Universidad Zaragoza

Proyecto Fin de Carrera

Ingeniería Informática

Configuración de servicios de red de un dispositivo empotrado mediante una base de datos

Autor

Pablo Gómez Duro

Director

Roberto Casas Millán

Ponente

Jesús Alastruey Benedé

Escuela de Ingeniería y Arquitectura

Mayo 2013

Configuración de servicios de red de un dispositivo empotrado mediante una base de datos

RESUMEN

El proyecto ha consistido en el desarrollo de un *backend* para una plataforma de enrutamiento basado en políticas, cortafuegos y control de navegación web que se instalará en un sistema empotrado. El origen del proyecto surge por la necesidad de integrar varios productos, desarrollados anteriormente por la empresa, en un mismo sistema. Dichos productos proporcionan funcionalidades de enrutamiento, cortafuegos, traducción de direcciones, redes privadas virtuales (VPN), calidad de servicio (QoS), proxy de navegación web y análisis de tráfico, además de otros servicios de red básicos como son la asignación de direcciones y resolución de nombres dentro de una red.

El trabajo realizado se ha centrado en proporcionar una abstracción de la configuración del sistema a través de una base de datos relacional para que posteriormente se desarrolle un frontal de administración web usando dicha base de datos como interfaz con el resto del sistema. Dicha abstracción se implementa mediante una base de datos diseñada para almacenar todo lo necesario para configurar el sistema, es decir, datos de configuración, reglas de filtrado de paquetes, reglas de enrutamiento, etc. Partiendo de los datos de configuración almacenados en la base de datos, se ha desarrollado el programa que genera la configuración adecuada para el sistema operativo y el software utilizado, y que es regenerada en el arranque del sistema o cada vez que se realice un cambio en la configuración.

Para la implementación de la plataforma se ha usado el sistema operativo Linux como base. Linux proporciona todas las herramientas necesarias para implementar la mayor parte de las funcionalidades de red y es lo suficientemente estable para un sistema de este tipo. El resto de servicios se han implementado usando otros proyectos de software libre como OpenVPN, Squid, Dnsmasq, LSM, Ntop, Collectd, etc.

Además de los servicios de red más usuales (enrutamiento, filtrado de paquetes o traducción de direcciones), se ha dotado al sistema de capacidades más avanzadas con el objetivo de mejorar la gestión del tráfico de red (aprovechamiento de ancho de banda, redundancia, tolerancia a fallos…), como por ejemplo:

- Monitorización y balanceo de tráfico dirigido hacia Internet entre varios enlaces.

- Agregación de varias conexiones VPN punto a punto enrutadas por varios enlaces.

- Balanceo de tráfico de entrada a nivel DNS.

Contenido

Índice de figuras

1 Objetivos y alcance del proyecto

1.1 Descripción

El proyecto ha consistido en el desarrollo de un sistema que facilite e integre la configuración de un dispositivo con funcionalidades de red. Estas funcionalidades incluyen servicios de red básicos, enrutamiento avanzado, balanceo y redundancia de tráfico, redes privadas virtuales (VPN) y monitorización y control de tráfico. Los requisitos iniciales eran que el sistema ofreciese una interfaz gráfica de configuración web que permitiese administrar de forma sencilla toda la configuración del sistema, de manera que una persona con conocimientos generales de redes no tuviese problemas en la operación del mismo. Dados estos requisitos se optó por dividir el sistema en dos partes bien diferenciadas, una que gestionaría toda la configuración de red, sistema operativo, etc., y otra que se limitaría a ser la interfaz con el administrador del sistema, como hemos dicho anteriormente a través de una interfaz web. El acoplamiento entre ambas partes del sistema se realiza a través de una base de datos que almacena la configuración del sistema. La parte de la interfaz de usuario introduce cambios en la base de datos y la parte del sistema reacciona ante estos cambios mediante un sistema de gestión de eventos.

Este proyecto fin de carrera ha abarcado las partes que comprenden la base de datos y los mecanismos de acceso a ella y la parte de configuración del sistema. La base de datos proporciona una abstracción de toda la configuración de red y servicios. La configuración contempla los parámetros básicos de interfaces de red, tablas de enrutamiento, reglas de filtrado y balanceo de tráfico, configuración de servicios VPN, control de navegación web, monitorización del sistema, etc. En la figura 1 se puede ver un esquema general del sistema desde el punto de vista de conectividad a redes.

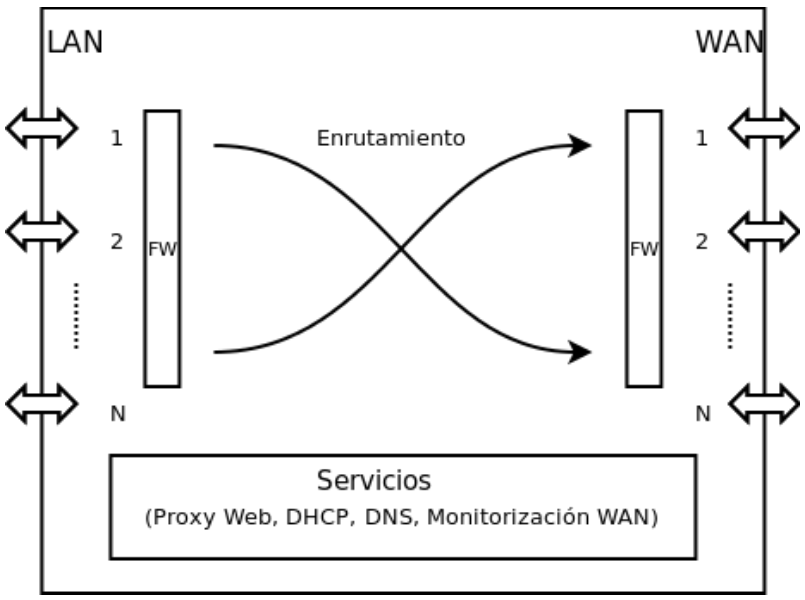

Figura 1: Esquema general del sistema

1.2 Contexto

El proyecto se ha realizado en la empresa Diaple Networking y comienza con la necesidad de integración de varios software para infraestructura de comunicaciones que la empresa instalaba en sus clientes. Todos estos software estaban diseñados para funcionar en máquinas independientes, pero la interacción entre los distintos software y los diferentes sistemas operativos sobre los que funcionaban hacían difícil una integración simple para su funcionamiento en una misma máquina. Aunque su integración no es imposible sería de dudosa y costosa mantenibilidad en el futuro.

Los productos a integrar eran los siguientes:

- NetConnect: Plataforma de enrutamiento y cortafuegos empresarial con capacidades de balanceo entre varias líneas y soporte para VPN. Basada en sistema operativo FreeBSD.
- MuliISP: Solución para la agregación de túneles VPN sobre varias líneas de acceso a internet. Basado en sistema operativo GNU/Linux.
- NetEye: Producto para control de navegación web. Basado en sistema operativo GNU/Linux.
- NetAnalizer: Producto para monitorizar el tráfico de la red. Basado en sistema operativo GNU/Linux.

El principal problema a la hora de integrar todos los productos es el sistema operativo. Estos sistemas tienen una dependencia muy alta del sistema operativo sobre el que se ejecutan ya que interaccionan con la configuración del subsistema de red, el cual difiere mucho de un sistema operativo a otro, a pesar de tratarse de sistemas Unix. Evidentemente se ha de elegir migrar todo a uno de los sistemas operativos.

Los productos más dependientes del sistema operativo son NetConnect y MultISP porque son los tienen interacción directa con el núcleo del sistema operativo. NetConnect funciona sobre el sistema operativo FreeBSD que no dispone de soporte para la agregación de túneles VPN, y por tanto esto nos deja sólo con la opción de usar GNU/Linux como sistema operativo base.

Otro problema a tener en cuenta en la integración es la forma en la que los sistemas guardan la configuración. Para simplificar y hacer más coherente el sistema final se optó por integrar toda la configuración con una pequeña base de datos con la intención de que más adelante se pueda desarrollar una interfaz de usuario para gestionar la configuración. La idea es que interfaz de usuario simplemente interactúe con dicha base de datos, abstrayéndose así de la configuración del sistema a más bajo nivel.

1.3 Trabajo previo

El trabajo desarrollado se basa enteramente en el uso de software libre. El software libre presenta múltiples ventajas ya que permite ahorrar costes en licencias y además presenta una gran robustez y seguridad al ser mantenido por un gran número de colaboradores.

Como sistema operativo se usa GNU/Linux, un sistema operativo que dispone de todas los requisitos de funciones de red necesarios para el proyecto. Además es un sistema

operativo muy extendido en el campo de aplicación del sistema desarrollado y por tanto eso nos garantiza fiabilidad y robustez en su funcionamiento. El núcleo del sistema operativo se encargará de hacer las tareas relacionadas con:

- Configuración de interfaces de red
- Enrutamiento
- Filtrado de paquetes.
- Traducción de direcciones (NAT)
- Calidad de servicio (QoS)

Para interactuar con dicha configuración Linux proporciona dos frameworks (iproute2 y Netfilter) para que herramientas de espacio de usuario puedan cambiar el comportamiento del sistema.

Para implementar el resto de servicios se han usado otros proyectos de software libre:

- OpenVPN para redes VPN
- Squid como proxy web
- Monitorización y análisis de red
    - Collectd: Gráficas de recursos del sistema
    - Ntop: Análisis de tráfico de red
- Dnsmasq para implementar el servicio DHCP y DNS local

2 Análisis

En este punto se detallan los pasos correspondientes al análisis del software implementado. Se analizan los requisitos iniciales que debía cumplir el producto en el momento de su definición. También se analizan las tecnologías usadas y la arquitectura adoptada para el sistema justificando las decisiones tomadas en estos aspectos.

2.1 Requisitos

Los requisitos iniciales que el sistema debía cumplir son los siguientes:

- Sistema de enrutamiento entre varias redes LAN y WAN sin restricción de número.
- Capacidad de filtrado de tráfico y traducción de dirección. Posibilidad de definir conjuntos de direcciones y puertos para facilitar la configuración.
- Capacidad de definir grupos de balanceo de tráfico de salida sobre las interfaces WAN.
- Soporte para interfaces de red compuestas (puentes y agregaciones) e interfaces virtuales.
- Calidad de servicio (QoS). Posibilidad de definir diferentes colas para distintos tráficos, asignándoles una prioridad y un ancho de banda máximo y mínimo.
- Redes Privadas Virtuales (VPN): Funcionamiento como cliente y servidor bajo el software OpenVPN. Capacidad de agregación de conexiones VPN sobre varios enlaces WAN para alta disponibilidad y mejora de ancho de banda.
- Servicio DHCP y DNS local.
- Integración de un proxy web para control de navegación web.
- Integración de toda la configuración de las funcionalidades anteriores en una base de datos y generación automática de la configuración consultando a dicha base de datos.

2.2 Tecnologías usadas

En este punto se describen las tecnologías que se han usado para la implementación del sistema.

2.2.1 Lenguaje de programación

Para configurar el sistema operativo y el software en función a la configuración presente en la base de datos se han programado scripts en lenguaje PHP que realizan consultas a la base de datos y generan automáticamente la configuración. La decisión de utilizar PHP en este entorno viene dada por la futura interfaz web que se desarrollará en este lenguaje. Aunque PHP es normalmente utilizado en entornos web y pueda resultar extraño usarlo para esta tarea, se trata de un lenguaje completo que no muestra ninguna carencia para este proyecto.

2.2.2 Base de datos relacional

Como base de datos se ha empleado SQLite. Una base de datos SQLite es una base de datos relacional especialmente diseñada para dispositivos empotrados que almacena toda la información de la base de datos en un fichero, de manera que no es necesaria ninguna configuración ni ningún servicio adicional ya que no hay conexión a la base de datos porque el programa que hace uso de la base de datos trabaja directamente sobre el fichero. Una ventaja de este tipo de configuración es que como toda la configuración queda integrada en la base de

datos, que como hemos dicho es un único fichero, podemos exportarla e importarla con facilidad para su uso en diferentes dispositivos, backups, etc.

2.2.3 Abstracción en el acceso a datos

Para la implementación de la persistencia de datos se ha utilizado un framework ORM (Object Relational Mapping) sobre PHP llamado Propel. La técnica ORM consiste en enlazar un modelo orientado a objetos con un modelo relacional. En este caso Propel realiza un mapeo de objetos PHP a un esquema SQLite. Esto facilita mucho el trabajo de inserción, modificación y consulta de la base de datos. Además tiene la ventaja de que una vez se tiene el modelo de objetos se puede extender para añadir funcionalidad a las clases de cada entidad de la base de datos.

Propel es uno de los framework ORM más utilizados en entornos PHP. Proporciona todas las funcionalidades típicas de este tipo de soluciones, patrón registro activo, validación, herencia, etc. Además es fácilmente extensible mediante unos componentes llamados *behaviors*, de los cuales se ha hecho uso para solucionar algunos de los problemas que se explicarán a continuación.

2.3 Planificación

En la figura 2 se observa la planificación establecida para el proyecto. En general la primera fase es de estudio de las necesidades y de las posibilidades para llevar a cabo el trabajo. La segunda fase de diseño e implementación comienza una vez se han obtenido conclusiones sobre cuáles van a ser las necesidades y sobretodo las posibilidades a la hora de su implementación.

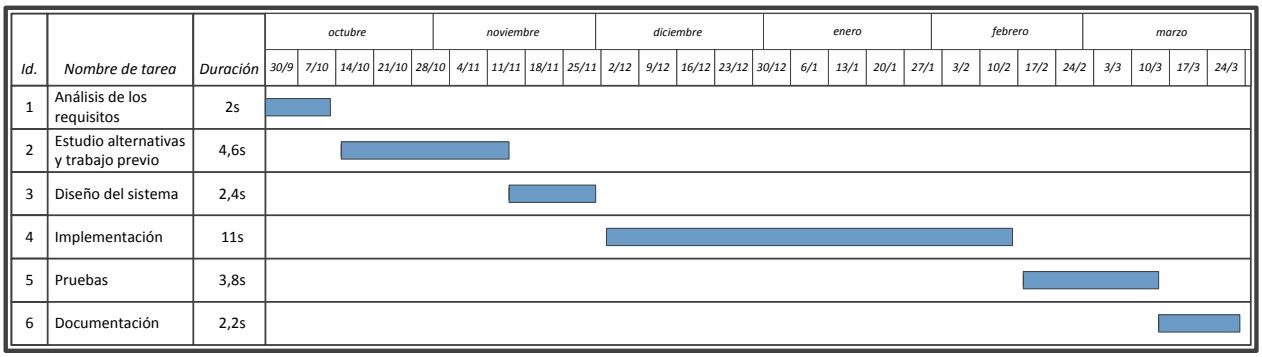

Figura 2: Diagrama de Gantt de la planificación

2.4 Arquitectura del sistema

En este punto se va a describir cuál es la arquitectura del sistema, sus componentes y la interfaz y relaciones entre ellos. No se profundizará demasiado en los detalles internos de ellos y se dejarán para puntos posteriores, centrándose en la interfaz entre los distintos componentes.

Como hemos comentado en el punto anterior, el esquema general del sistema es por un lado un módulo que introduce datos en una base de datos, al que llamaremos Frontend. Por otro lado habrá un módulo, al que llamaremos Backend, que lee dichos datos y genera la

configuración del sistema operativo y los servicios. Por generar la configuración entendemos ejecutar comandos, generar ficheros de configuración, parar o arrancar procesos, etc. La interfaz entre ambos, como veremos más adelante, también la vamos a considerar un módulo por la funcionalidad que implementa. A este módulo lo llamaremos Core. En la figura 3 se muestran dichos módulos y el flujo de datos entre ellos.

De esta manera se intenta hacer que el acoplamiento entre ambas partes sea la mínima y se limite al modelo de datos. No obstante, ha sido necesario desarrollar un mecanismo que permita notificar al Backend cuando haya una modificación en la base de datos que afecte a la configuración del sistema. Este mecanismo de notificación lo implementa el Core. Para implementarlo se ha hecho uso de las capacidades del framework de acceso a la base de datos utilizado, en el punto 3 hay más en detalle de esto. En la figura 3 se puede ver cuál es el flujo de datos en ese caso.

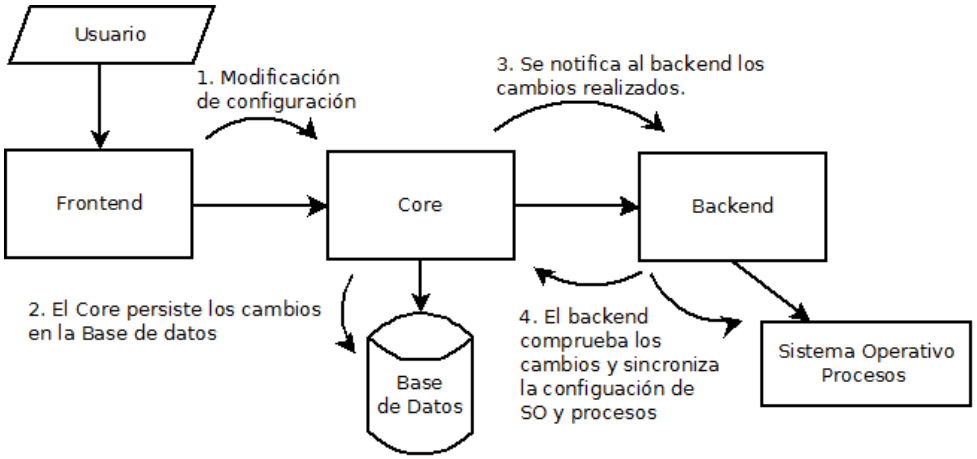

Figura 3: Diagrama de módulos del sistema 2

Como se puede ver el trabajo desarrollado comprende dos partes bien diferenciadas. Una la del desarrollo del todo el modelo de datos, gestión de configuración, eventos y notificaciones. Esta parte será fundamentalmente un programa escrito en PHP explotando al máximo la orientación a objetos para conseguir un sistema lo más modular posible. Por otro lado está la parte más cercana a la configuración de red y del sistema operativo. Esta parte ha consistido, en primer lugar, en investigar todas las soluciones posibles para conseguir ofrecer los servicios de red necesarios y posteriormente en integrar la solución elegida con el programa de gestión de la configuración, que se dedicará a generar todos los ficheros y comandos de sistema necesarios para configurar el sistema operativo y el software adicional empleado.

Como hemos dicho, se ha implementado el módulo de abstracción para que ni Backend ni Frontend tengan que conocer sus detalles de implementación, haciendo por tanto el sistema mucho más mantenible y comprensible. Esta interfaz consiste en una parte común de acceso a datos y luego un sistema de eventos que permite que ambas partes pueden ser notificadas correctamente cuando una de ellas modifica algún dato o se produce un evento que requiera tratamiento. De ésta manera conseguimos el mínimo acoplamiento posible del que hablábamos anteriormente.

3 Módulo de comunicación y abstracción (Core)

Como hemos mencionado en el punto anterior el Core es la capa que implementa la interfaz entre el Frontend y el Backend del sistema. Este módulo es el encargado de abstraer la configuración de red en un modelo de datos que contempla la información estrictamente necesaria para configurar el sistema. Sus funciones son las siguientes:

- Proporcionar al Frontend una forma de modificar la configuración mediante un modelo de datos totalmente abstracto al sistema operativo y el software utilizado.
- Persistencia de la configuración.
- Notificar al Backend de los cambios que se realicen en la configuración.
- Permitir la aplicación diferida y atómica de los cambios de configuración realizados.

Además de todas estas funcionalidades se ha optado por subdividir el Backend en Servicios. Cada servicio se encargará de una parte del sistema y se podrá habilitar o deshabilitar independientemente del resto, teniendo en cuenta siempre sus dependencias de otros módulos. Esto permite que el sistema final se pueda personalizar incluyendo sólo los módulos necesarios.

A continuación se describen con más detalle las características de este módulo así como su implementación.

3.1 Esquema Entidad Relación del modelo de datos

En el anexo A está disponible el diagrama entidad-relación (DER) y el esquema relacional completo de la base de datos. La base de datos se irá describiendo poco a poco en cada uno de los servicios de la sección Backend. Para cada servicio se describirá que datos son necesarios y que entidades y relaciones han sido creadas para tal efecto.

3.2 Bases de datos de configuración y tiempo de ejecución

Una de las particularidades del sistema es que la base de datos representa en cada momento una configuración del sistema operativo para que preste los servicios de red. Por tanto cualquier modificación que hagamos en la base de datos implicará un cambio de configuración de los servicios de niveles inferiores para adaptase a la nueva configuración. Esto puede suponer un problema si deseamos realizar varios cambios en la base de datos pero queremos que se apliquen de forma atómica. Por ejemplo, queremos modificar un conjunto de reglas del firewall, es posible que uno de los cambios nos provoque un corte de comunicación con el sistema y no podamos aplicar el resto de reglas. Es por esto que tiene que ser posible retrasar la aplicación de un conjunto de cambios para así poder asegurarnos de que los cambios que se aplican son los que realmente deseamos.

Hay varias alternativas a la hora de implementar esta funcionalidad, una de ellas es delegar en el Frontend el almacenamiento temporal de los cambios realizados. De esta manera cuando un el administrador decida aplicar los configuración el Frontend pasará a la base de datos los cambios que se hayan realizado. Para evitar tener que lidiar con estos temas en el Frontend se ha implementado otra alternativa para esta funcionalidad. La solución implementada consiste en tener dos copias idénticas de la base de datos (a nivel de esquema), una sobre los que el Frontend aplica los cambios y otra sobre la que el Backend lee la

configuración. La sincronización entre ellas se lleva a cabo mediante la capa de abstracción en acceso a datos (ORM) y es transparente para ambos módulos. De esta manera conseguimos una mayor modularización y facilita mucho la creación de nuevos frontends o backends, ya que su única interfaz es la base de datos y no existe ningún acoplamiento directo entre ellos.

3.3 Sistema de eventos

El sistema de eventos es el encargado de comunicar al Backend todos los cambios que se produzcan en la configuración. Como el sistema esta subdividido en módulos cada uno se encarga de un conjunto de datos de la configuración. Cada registro de la configuración tiene especificado un evento que es lanzado cuando este se modifica. El sistema de eventos proporciona a los servicios la manera de registrarse para que sean avisados cuando se produzca un evento. Dicho registro consistirá en que el servicio especifique un evento y un método que será invocado cuando se produzca dicho evento. El método recibirá como parámetro el registro modificado y deberá realizar las acciones necesarias para mantener la configuración de red sincronizada con lo que refleja la base de datos.

3.4 Implementación

Tal y como hemos dicho, el Backend está dividido en servicios. Cada servicio implementa una funcionalidad común que permite tratarlos a todos de similar forma. Esto se ha implementado haciendo que todos los servicios hereden de una clase abstracta que proporciona la funcionalidad común y la interfaz que deben implementar todos los servicios. Esta abstracción proporciona la siguiente funcionalidad común:

- Control de dependencias entre servicios.
- Escucha y generación de eventos entre servicios.
- Inserción de configuración en otros servicios (*hooks*).

Todos los servicios heredarán dicha funcionalidad común, únicamente deberán implementar los métodos de configuración, arranque y parada. La figura 4 muestra el diagrama de clases.

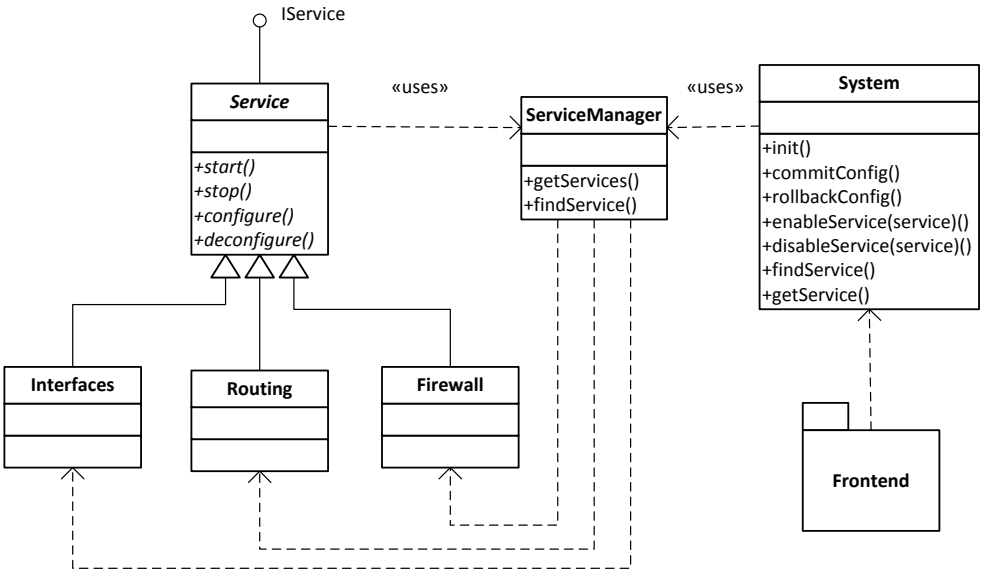

Figura 4: Diagrama de clases

Como se observa en la figura 4, en el diagrama se ha introducido la clase ServiceManager que es encargada de controlar todos los servicios del sistema. Mantiene referencia a ellos para que sea posible la comunicación entre servicios. Cuando un servicio necesite de alguna funcionalidad de otro servicio, resolverá la dependencia a través de la clase ServiceManager. También tenemos la clase System que ofrece funciones de arranque y parada global y métodos para controlar el estado de los cambios de configuración pendientes. El objetivo de la clase System es que sea el punto de acceso desde el Frontend, siguiendo el patrón de diseño Fachada o *Facade.*

4 Backend

Como hemos dicho anteriormente el Backend es la parte del sistema que interactúa finalmente con la configuración de red. Recibirá eventos del módulo de abstracción cuando haya cambios y se encargará de mantener la configuración de red sincronizada.

Se ha dividido en servicios para hacer el producto final más personalizable. Cada servicio corresponde a una implementación de la clase abstracta Service nombrada anteriormente. Dicha implementación debe contener métodos para inicializar y parar el servicio y podrá registrar métodos que serán ejecutados cuando la base de datos genere un evento determinado.

Lógicamente, aun dividiendo en módulos el sistema, hay módulos que dependen de otros para llevar a cabo su funcionalidad. Por ello en la definición de cada módulo se especifican sus dependencias tal y cómo se ha explicado en el punto 3.5.

A continuación se van a detallar todos los servicios del sistema. Para seguir una pauta común, en cada servicio se va a explicar la funcionalidad requerida, los datos necesarios para implementar dicha funcionalidad, las dependencias de otros servicios, y finalmente las alternativas de implementación y la solución final adoptada. El proceso de configuración, con los comandos y ficheros generados, se especifica para cada servicio en el anexo B.

4.1 Interfaces

El servicio de interfaces es el servicio más importante del sistema, ya que la mayoría del resto de servicios depende de él. Este servicio nos va a permitir definir y configurar las interfaces de red por las que van a entrar y salir paquetes de datos del sistema.

Las interfaces de red corresponden con las tarjetas de red (NIC) que tiene físicamente un sistema. Cada interfaz de red puede recibir y enviar tráfico y tiene una configuración IP asociada, que es la que le permite determinar qué tráfico debe recibir. El sistema operativo recibe el tráfico y mediante el proceso de enrutamiento determina cuál es su interfaz de salida, como se puede observar en la figura 5.

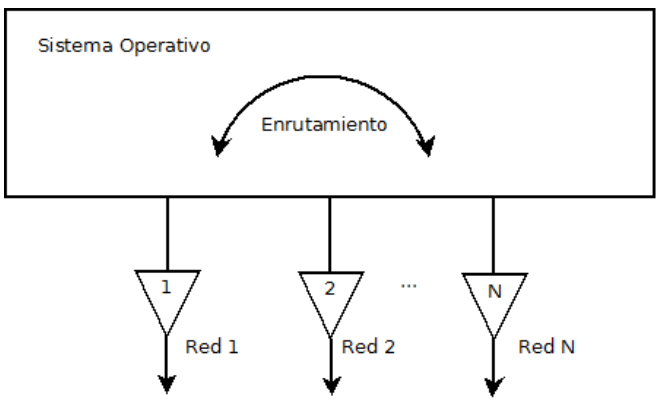

Figura 5: Esquema general de interfaces

Las interfaces de red pueden ser agrupadas para formar interfaces virtuales que ofrezcan funcionalidades adicionales. Estas agrupaciones pueden ser de cuatro tipos:

- Puente o Bridge
- Agregación (Bonding en Linux)
- Interfaces VLAN
- Interfaces Virtuales

El funcionamiento de estas interfaces especiales es totalmente transparente para el sistema, es decir, el sistema operativo hace uso de estas interfaces de la misma manera que si fueran interfaces físicas.

Lógicamente puede interesar combinar estos tipos de interfaces y es por ello que el modelo de datos contempla relacionar las interfaces en forma de árbol para poder abarcar todas las posibilidades. Desde un punto de vista más formal el modelo de datos implementa un DAG [1](Directed Acyclic Graph o Grafo Acíclico Dirigido). La figura 6 muestra un ejemplo de lo que podría ser un conjunto de interfaces:

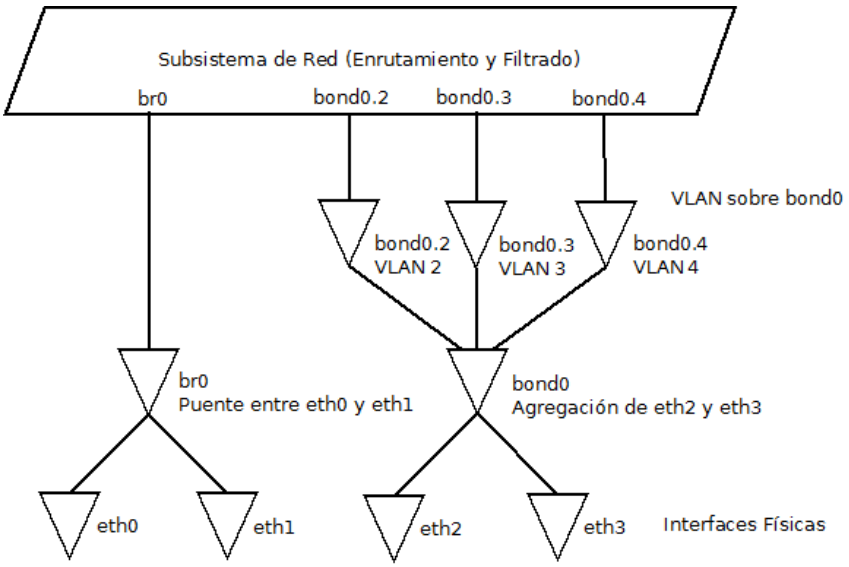

Figura 6: Esquema de composición de interfaces

Las interfaces finales utilizadas por el proceso de filtrado y enrutamiento serán las que se encuentran en la parte superior del árbol, y las interfaces físicas las que se encuentran en la parte inferior. Como hemos dicho, en el nivel superior no hay diferencia entre la configuración de los diferentes tipos de interfaces. Estas interfaces son las que recibirán la configuración IP. De aquí en adelante llamaremos interfaces lógicas a estas interfaces para distinguirlas de las interfaces físicas.

La configuración de una interfaz supone la dirección IP, la máscara de subred y el gateway en caso de que lo haya. La máscara de subred determinará la subred a la que la interfaz pertenece, es decir, las IP que son alcanzables directamente a través de la interfaz configurada. La presencia o no de gateway determinará si se trata de una interfaz LAN o WAN

[1] http://en.wikipedia.org/wiki/Directed_acyclic_graph

para el sistema. Si tiene gateway consideraremos que esa interfaz sirve para dar acceso a internet y por tanto será de tipo WAN. En cambio, si no se ha configurado un gateway se considerará que la interfaz es de tipo LAN, ya que todo el tráfico recibido por dicha interfaz tendrá que ser enrutado por otra de las interfaces presentes en el sistema.

Realmente este modelo de configuración de interfaces en árbol es tan general que permite configuraciones incoherentes, como por ejemplo encadenar varias interfaces VLAN con tags distintos (la primera eliminará el tag, las demás no harán nada). De todas formas se ha preferido dejar esas comprobaciones de situaciones incoherentes a otro nivel y permitirlas en el modelo.

4.1.1 Tipos de interfaces

A continuación se explican con más detalle las diferentes configuraciones que pueden tener las interfaces del sistema:

4.1.1.1 Puente entre interfaces (Bridging)

Las interfaces agrupadas en un puente están unidas a nivel de enlace, es decir, que todas las tramas de nivel de enlace que lleguen por una y no tengan como destino la propia máquina serán redirigidas por todas las demás interfaces. Las tramas de datos retransmitidas atraviesan el sistema sin llegar al nivel de red, comportándose como un switch. La consecuencia de este comportamiento es que los paquetes no se verán afectados por ninguna regla de filtrado IP, por tanto no es posible usar dichas reglas para filtrar tráfico. A este nivel, en cambio, podemos hacer filtrado en función a la dirección MAC que tengan las tramas Ethernet.

4.1.1.2 Agregación de interfaces (Bonding)

La agregación de interfaces o bonding (nombre que se le da en Linux) es una funcionalidad que proporciona el sistema operativo que permite unir varias interfaces físicas en una interfaz virtual con capacidades de alta disponibilidad. Es decir, el sistema podrá enviar paquetes por la interfaz virtual pero realmente esos paquetes serán enviados por las demás interfaces según sea la configuración. Esta agregación ofrece varios modos de funcionamiento, dependiendo de cómo envía los paquetes el sistema. Por un lado, una de las posibilidades es configurar la agrupación en modo activo-pasivo, que hará que sea una de las interfaces la que esté transmitiendo y recibiendo información y el resto estén a la espera por si la primera fallase. Por otro lado ofrece varias posibilidades para configurar la agrupación como activo-activo. Una de ellas permite que la transmisión de paquetes sea vaya alternando entre las interfaces (Round Robin). Esto nos permite aumentar el ancho de banda de transmisión pero no así el de recepción, porque los paquetes seguirán llegando sólo por una de las dos interfaces. Si queremos balanceo en la recepción de tráfico también se ofrece un modo que habilita el protocolo LACP, que negocia la conexión con los switches a los que está conectado. En este caso, lógicamente, será necesario hardware de red especial que soporte dicho protocolo.

4.1.1.3 Interfaces VLAN

Las redes VLAN permiten el uso de varias subredes en un mismo segmento de red como si fuesen redes independientes. Esto se consigue añadiendo un campo a las tramas de nivel de enlace (Ethernet) tal y como especifica el estándar IEEE 802.1Q. Ese campo contiene un

identificador de red (VLAN tag) que permite diferenciar las tramas de una red y las de otra. Aplicado a nuestro caso quiere decir que el sistema operativo nos va a permitir filtrar tráfico según un determinado identificador. Es decir, podemos usar una misma interfaz física del sistema para tener varias interfaces lógicas que serán las que realmente use el sistema para el enrutamiento de paquetes.

4.1.1.4 Interfaces virtuales

Las interfaces virtuales permiten definir interfaces a las que procesos de usuario se pueden conectar para implementar las funciones de envío y recepción de datos. Su principal uso es para implementar redes privadas virtuales (VPN). Es como si tuviésemos una interfaz en la en vez de conectar un cable conectamos un proceso dentro del mismo sistema. Este proceso se encarga de recibir y enviar todo el tráfico que pasa por la interfaz. En Linux existen dos tipos de estas interfaces, dependiendo de si emulan un nivel de red (tun) o un nivel de enlace (tap).

Su uso con otras interfaces en la jerarquía permite opciones interesantes y muy versátiles como por ejemplo incluir una interfaz virtual conectada a una conexión VPN punto a punto y a su vez incluida en un puente con otra interface física. Si al otro extremo de la VPN tenemos la misma configuración nos serviría por ejemplo para extender una red LAN (usando una misma subred) de una empresa entre varias sedes en lugares distintos.

4.1.2 Datos de configuración

Como hemos dicho es importante distinguir entre las interfaces lógicas y las interfaces físicas, ya que quién va a recibir una configuración IP van a ser las interfaces lógicas. Por ello se ha optado por separarla en dos entidades distintas. Por un lado la entidad Interface representa la configuración de las interfaces lógicas y por otro la entidad Device representa las interfaces físicas y sus relaciones entre sí para representar el DAG nombrado anteriormente. De acuerdo con esto en la figura 7 se muestra el esquema de entidades.

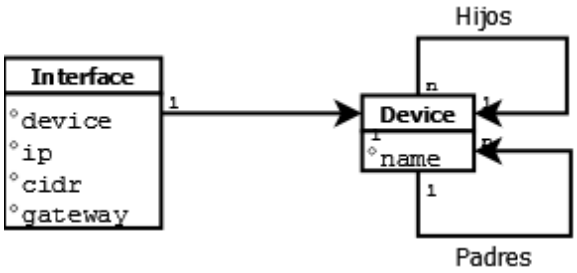

Figura 7: Entidades configuración de interfaces

4.2 Firewall

El servicio de Firewall proporciona la capacidad de seleccionar flujos de tráfico de determinadas características para aplicar sobre ellos acciones determinadas. Las acciones básicas que se aplican son para determinar si el tráfico es permitido o no, pero también es posible clasificar dicho tráfico de forma que otros servicios puedan tomar decisiones distintas en función de dicha clasificación. En este punto nos vamos a centrar en las formas de seleccionar el tráfico y cómo aceptarlo/rechazarlo o clasificarlo. En puntos siguientes, enrutamiento y calidad de servicio, se usará la funcionalidad de clasificación para tratar a los flujos de tráfico de forma adecuada.

El filtrado de paquetes actúa sobre la entrada de paquetes al sistema por cada una de la interfaces de red. Es importante decir que no actúa sobre las interfaces físicas, sino sobre las interfaces lógicas, tal y como las hemos clasificado anteriormente. Actuar sobre las interfaces físicas no tiene sentido excepto en un caso, detallado más adelante, que será el filtrado en función de la dirección MAC de los paquetes. No tiene sentido porque precisamente lo que pretendemos creando interfaces lógicas es tener más control sobre el tráfico. Si actuásemos sobre la interfaces físicas estaríamos perdiendo toda la capacidad de filtrar sobre agregaciones de interfaces o VLAN distintas.

Una vez un paquete ha atravesado el filtrado de la interfaz por la que entra al sistema, pasará a la fase de enrutamiento y saldrá por una de las interfaces restantes sin ningún tipo de restricción. Es decir, los paquetes no son filtrados a la salida del sistema sino a la entrada. Esto implica que el tráfico local generado por el propio sistema no será, en principio, objeto de ningún tipo de restricción.

El modelo de firewall implementado se basa en reglas que permiten seleccionar el tipo de tráfico y aplicarle una determinada acción, que será aceptar o rechazar el tráfico. Las reglas se aplican en orden y su aplicación termina cuando el tráfico coincide con lo especificado en una de las reglas, es decir, que la primera regla que coincida con el tráfico será la que se aplique.

Para aumentar la seguridad, si un determinado tráfico no coincide con ninguna regla el tráfico será rechazado. De esta manera se evita posibles fallos de seguridad debido a algún error o descuido a la hora de introducir las reglas de filtrado en la configuración.

Cuando hablamos de selección de tráfico nos referimos a seleccionar paquetes o flujos (TCP, UDP, etc.) en función de datos de la cabecera IP o del protocolo superior, es decir, datos como las direcciones de origen o destino y los puertos origen o destino. También es posible realizar la selección en función a un grupo de IP o puertos y negar las condiciones. Más adelante veremos la estructura de datos necesaria en base de datos para guardar la información que permita implementar estas reglas.

El servicio también contempla la definición de reglas NAT (Network Address Translation) que nos permitirán hacer traducción de direcciones siguiendo las mismas condiciones de selección de tráfico. Las traducciones posibles son en la dirección origen y en la dirección destino, Source NAT y Destination NAT respectivamente. DNAT nos permite por ejemplo hacer que los paquetes provenientes de internet sean redirigidos hacia una IP interna. SNAT nos permite que puestos de una red con direccionamiento privado puedan tener acceso a internet traduciendo su dirección origen a una dirección IP de direccionamiento público, o a una IP de direccionamiento privado que está en una red en la que el router de nivel superior saber enrutar hacia ella.

4.2.1 Sistema de filtrado de paquetes en Linux

El subsistema de filtrado en Linux recibe el nombre Netfilter. Netfilter provee todo la funcionalidad del núcleo del sistema operativo relativa a filtrado y traducción de paquetes de red y las herramientas de espacio de usuario para interactuar con su configuración. La

herramienta utilizada para configurar las reglas es el comando Iptables y esta implementada sobre Netfilter.

Iptables permite generar un conjunto de reglas que nos permiten implementar el comportamiento deseado para nuestro sistema. Se basa en reglas divididas en una parte de selección y otra de acción. Las posibilidades que ofrece son tan amplias que se puede implementar muchas funcionalidades de filtrado y clasificación de tráfico e incluso, en algunos casos, de varias maneras cada una de ellas.

Las reglas se agrupan en cadenas, que son conjuntos ordenados de reglas. A su vez las cadenas se agrupan en tablas según el propósito de las reglas que albergan. Existen unas cadenas predefinidas para cada uno de los estados por los que pasa un paquete en su tránsito por el sistema. Estas cadenas son las siguientes:

- PREROUTING: Atravesada por los paquetes antes de que se tome la decisión sobre su enrutamiento, es decir por qué interfaz de red debe salir el paquete o si es local.
- INPUT: Atravesada por los paquetes con destino local.
- FORWARD: Atravesada por los paquetes con destino no local.
- OUTPUT: Atravesada por los paquetes que tienen como origen el propio sistema.
- POSTROUTING: Atravesada por todos los paquetes que salen del sistema una vez tomada ya la decisión de enrutamiento, es decir la interfaz de salida es ya conocida.

Como hemos dicho las cadenas se agrupan en tablas de la siguiente manera:

- Tabla filter: Contiene reglas cuyo objetivo es decidir sobre si un paquete es filtrado o no, es decir, si permitimos que continúe su paso por el sistema o lo descartamos. Contiene las cadenas INPUT, FORWARD y OUTPUT.
- Tabla nat: Contiene reglas cuyo objetivo es aplicar mecanismos de traducción de direcciones para transformar las direcciones de los paquetes y así poder redirigirlos. Contiene las cadenas PREROUTING, OUTPUT y POSTROUTING.
- Tabla mangle: Contiene reglas cuyo objetivo es modificar parámetros de los paquetes, tales como el tipo de servicio (TOS) o hacer marcado para tomar otras decisiones más adelante en el procesamiento de paquete. Contiene todas las cadenas mencionadas anteriormente.

La figura 8 muestra cuál es el recorrido de un paquete y como va atravesando las cadenas de reglas desde que entra en el sistema hasta que sale de él.

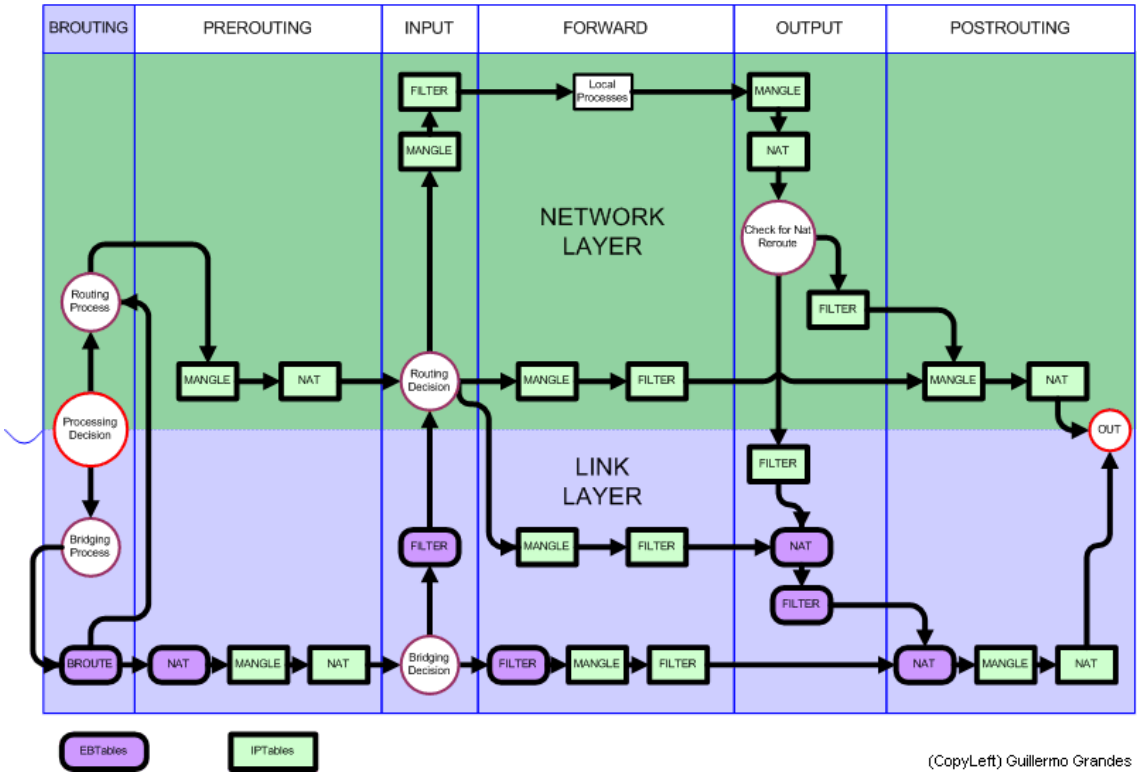

Figura 8: Diagrama del flujo de paquetes por netfilter en Linux

El funcionamiento de las cadenas de reglas consiste en que el paquete va recorriendo las reglas en orden y cuando llega a una regla en la que se cumple la condición de la regla se ejecuta la acción asociada. Hay dos tipos de acciones: acciones terminales y no terminales. Una acción terminal hace que se dejen de procesar las reglas restantes de la cadena, mientras que una acción no terminal hacer justamente lo contrario. Ejemplos de acciones terminales son ACCEPT y DROP que son las acciones usadas para aceptar y rechazar un paquete respectivamente. Como ejemplo de acción no terminal podemos nombrar la acción MARK, que permite marcar un paquete y continuar normalmente con su procesamiento, o también la acción LOG, que permite registrar el paso del paquete en el log del sistema.

También es posible definir cadenas personalizadas y usar su nombre como acción para redirigir a ellas. En este caso si se desea interrumpir su procesamiento se podrá salir de ellas usando la acción RETURN.

4.2.2 Filtrado en capa de aplicación (Application-layer Firewall)

Las reglas de filtrado que manipula iptables sólo afectan al nivel de red y transporte pero no es posible hacer selección de paquetes en función del protocolo de nivel de aplicación (capa 7).

Para realizar este tipo de filtrado es necesario o bien realizar modificaciones en el núcleo Linux, ya que por defecto no incluye soporte para este tipo de filtrado, o bien utilizar una aplicación en espacio de usuario a la que el sistema operativo pasará los paquetes para que su contenido sea inspeccionado. Esta última opción implica el uso de un mecanismo de Netfilter

que permite la comunicación entre el núcleo del sistema operativo y un proceso de espacio de usuario.

La aplicación encargada de analizar los paquetes es l7-filter que hace un marcado de paquetes cuando identifica un protocolo de nivel de aplicación, de manera que posteriormente se pueda usar la marca del paquete con las reglas de iptables para tomar la acción deseada.

La implementación de este tipo de filtrado mediante un proceso de espacio de usuario requiere especial cuidado ante posibles fallos de dicho proceso, ya que si el proceso deja de estar en ejecución el sistema quedará completamente inoperativo ya que todo el tráfico de paquetes pasa a través del proceso.

4.2.3 Filtrado MAC

El filtrado por dirección MAC permite establecer reglas que filtren determinadas tramas Ethernet en función de su origen. En Linux el filtrado MAC pude hacerse a dos niveles distintas. Lo normal es hacer el filtrado MAC a nivel de enlace, pero esa opción sólo está disponible si trabajamos con interfaces de tipo puente. En caso de trabajar con interfaces puente podemos, no sólo filtrar en el puente, sino que podemos hacer filtrado específico para cada uno de los puertos del puente. En caso de trabajar con interfaces normales, los paquetes no atraviesan el filtrado de nivel de enlace, y por tanto tenemos que usar los filtros IP para hacer el filtrado. En Linux los filtros en interfaces bridge se configuran con el comando *ebtables*. De las muchas opciones que ofrece nos centraremos en las funciones de filtrado básico, es decir dada una MAC y un puerto del bridge comprobar si aceptamos o denegamos ese tráfico. También permitiremos establecer la política del puerto de bridge. En el caso de querer filtrado MAC en interface no bridge, recurriremos a *iptables*, el cuál dispone de un módulo de selección que comprueba las direcciones MAC del paquete de forma similar a como lo haríamos para filtrar por direcciones IP.

4.2.4 Datos de configuración

Como hemos comentado anteriormente una regla de firewall va a estar asociada a una interfaz y tendrá que tener los siguientes campos para poder establecer las condiciones de selección y acción:

- Acción de la regla.
- IP origen/destino y puerto origen/destino. Tanto textualmente como referencia a un alias.
- Capacidad para especificar la inversión de cualquiera de los cuatro campos anteriores.
- Protocolo de nivel de red y de nivel de aplicación.

Para implementar el sistema de Alias se necesita una entidad que guarde los propios alias y otra con sus valores. Por tanto el modelo queda como se aprecia en la figura 9.

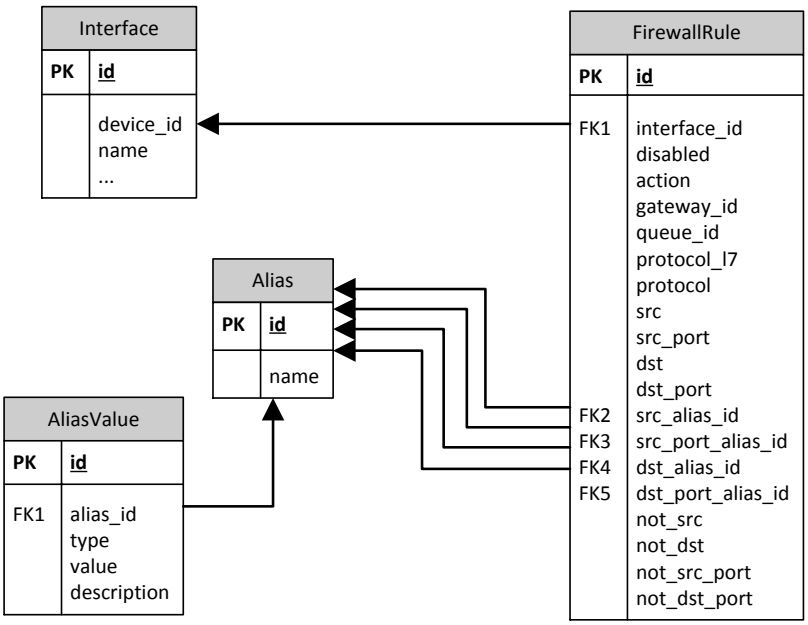

Figura 9: Entidades configuración de Firewall

Tal y como hemos dicho las reglas de filtrado también permitirán clasificar el tráfico para su enrutamiento y calidad de servicio. En los puntos correspondientes de dichos servicios se verá cómo encaja con las reglas de Firewall.

En cuanto a las reglas NAT las condiciones de selección serán similares, únicamente deberemos añadir algunos campos en cada caso:

- DNAT:
  - IP a la que traducir la IP destino.
  - Puertos externos e internos para hacer el mapeo.
  - Capacidad para especificar si se además de hacer la traducción se debe autorizar el tráfico automáticamente.
- SNAT:
  - IP a la que traducir la IP origen.
  - Puerto al que traducir el puerto origen.
  - Especificar si el puerto debe ser estático (no cambiado en la traducción).

Quedando el modelo según se ve en la figura 10.

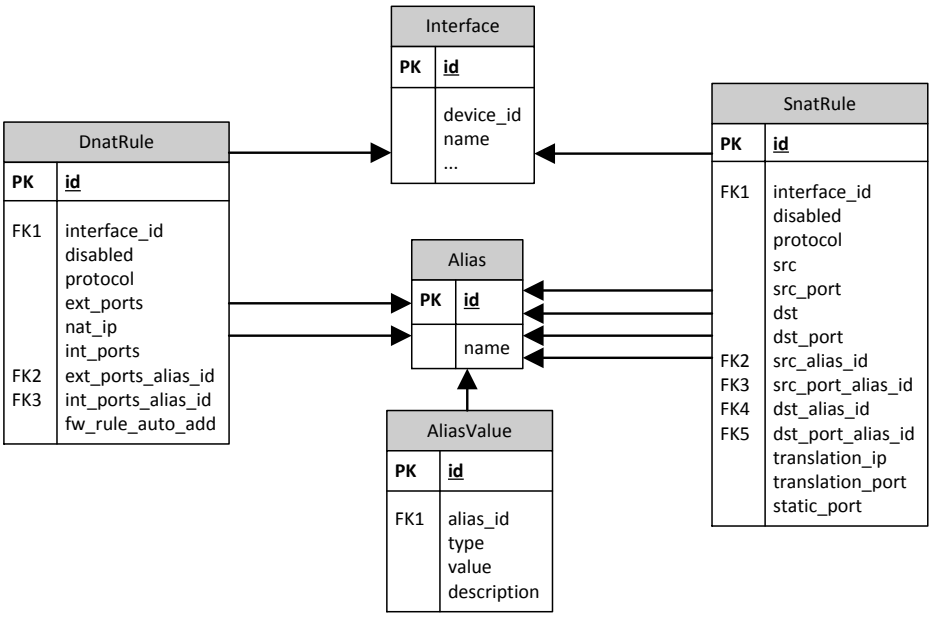

Figura 10: Entidades configuración de NAT

Para guardar la configuración del filtrado MAC se ha decidido utilizar una tabla para almacenar las reglas que referencia a la tabla de dispositivos físicos (Devices) para cada regla (ver figura 11). Como hemos dicho anteriormente las reglas MAC afectan al nivel de enlace por tanto es más adecuado referenciar a la tabla Devices en vez de a Interfaces ya que, por ejemplo, un puerto de un bridge jamás va a aparecer en la tabla Interfaces. Para gestionar la política de cada dispositivo se ha añadido un campo a la tabla Devices que nos indica si por defecto hay que rechazar o denegar el tráfico.

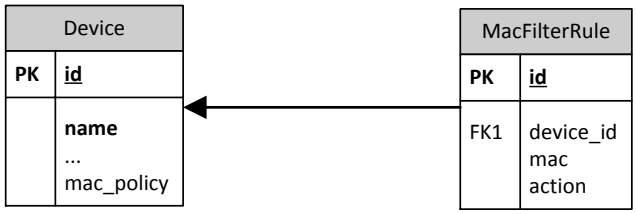

Figura 11: Entidades configuración filtrado MAC

4.3 Enrutamiento

El servicio de Firewall se encargaba de filtrar los paquetes que atravesaban el sistema, pero no tomaba ninguna decisión sobre cuál iba a ser el camino tomado por ese paquete un vez se ha autorizado su transmisión. Cuando el sistema recibe un paquete de datos necesita saber por dónde tiene que salir el paquete del sistema. De ello se encarga el proceso de enrutamiento.

En general un sistema puede tener varias interfaces de red con gateways configurados, eso significa que cualquiera de ellas es un camino de enrutamiento para los paquetes que atraviesan el sistema. Tradicionalmente el enrutamiento de paquetes se ha hecho en función de la dirección destino de estos. Con el tiempo surgió la necesidad de enrutar en función a otras características del paquete, como por ejemplo, dirección origen, protocolo, puertos, tipo de servicio, etc. Esto se conoce como Enrutamiento basado en políticas o Policy-Based Routing (PBR). Además de esto también existen técnicas para hacer que haya múltiples caminos posibles para enrutar, es decir, que sea posible balancear el tráfico entre varios enlaces buscando redundancia o maximizar la tasa de transferencia.

En este caso el servicio de enrutamiento va a implementar PBR basándose en las reglas de selección de tráfico del servicio de Firewall y también capacidades de balanceo entre varios enlaces. En los siguientes puntos se explica las posibilidades que ofrece Linux al respecto y la solución final adoptada.

4.3.1 Enrutamiento en Linux

Como se ha dicho, tradicionalmente el enrutamiento se ha hecho únicamente en función a la dirección destino de un paquete. En Linux esto se configura usando la tabla de enrutamiento, que consiste en una serie de rutas que especifican para un cierto destino que interfaz es la adecuada para alcanzarlo. En la tabla de rutas aparecerán las rutas de las redes locales que tiene el sistema, reglas específicas que fuercen a un rango de destino a enrutarse por cierta interfaz y finalmente una ruta por defecto que especificará la interfaz por la que se envían los paquetes en última instancia si el destino no coincide con ninguna de las rutas.

La tabla de rutas única es válida hasta que aparece el PBR. Para implementar PBR sobre Linux se añade una funcionalidad nueva, las reglas de enrutamiento. Como se ha explicado las rutas se agrupan en una tabla, la idea de las reglas de enrutamiento es que en el sistema pueden existir varias tablas de enrutamiento y la elección de una u otra será determinada por las reglas de enrutamiento. Gracias a esta funcionalidad es posible crear tantas tablas de enrutamiento como queramos y dirigir el tráfico a ellas según las características de cada flujo de tráfico.

4.3.2 Balanceo

El sistema requiere la posibilidad de balancear el tráfico entre las interfaces de salida para así obtener redundancia o mejor ancho de banda global. La salida de un paquete de datos por una de las interfaces de red implica que saldrá hacia internet con la IP pública del modem que haya detrás, por tanto, los paquetes de una mismo flujo de datos, ya sea una conexión TCP o un flujo UDP, ICMP, etc. deberán ser enrutados por la misma interfaz de red. Esto quiere decir que el balanceo de tráfico será por tanto a nivel de transporte y no a nivel de red, lo que

significa que no ganaremos en ancho de banda en una única conexión, sólo se podrá aprovechar todo el ancho de banda con múltiples conexiones simultáneas.

4.3.2.1 Alternativas

Linux ofrece un mecanismo llamado rutas multipath, que permiten especificar más de un gateway para la ruta. De esta manera los paquetes se balancean entre todos los gateways especificados en la ruta. El problema de esta solución es que el balanceo se realiza a nivel de red, es decir, que cada vez que el sistema enruta un paquete se toma la decisión sobre cuál es el Gateway de salida. Esto provoca que paquetes de una misma conexión (TCP, UDP, etc.) puedan salir por distintos gateways cada vez y como estamos trabajando en un entorno en el que hay traducción de direcciones eso provocaría que los paquetes llegasen a destino con direcciones origen distintas. Por tanto debemos emplear una solución que haga un balanceo a nivel de transporte y no a nivel de red.

Para implementar un balanceo a nivel de red se puede optar por usar reglas iptables para marcar los paquetes y posteriormente enrutarlos en función a esa marca. Cada Gateway de salida tendría una marca asociada y una tabla de enrutamiento y habría que configurar reglas de enrutamiento que dirigiesen los paquetes marcados a la tabla de rutas apropiada. Además como iptables tiene capacidad para relacionar un paquete a una conexión podremos guardar dicha marca en la conexión y así asegurarnos de que todos los paquetes de una misma sesión son etiquetados con la misma marca y por tanto enrutados hacia un mismo Gateway. Para conseguir balanceo de conexiones entre varios gateways se puede usar uno de los módulos estadísticos de iptables para asignar una marca aleatoria entre varias posibles.

En el caso del balanceo de tráfico se deberá prestar especial atención al estado de los enlaces, ya que si no tienen conectividad parte del tráfico se perderá. Para ello se usa el servicio de monitorización de enlaces (4.9.2) y al recibir un evento de cambio de estado de un enlace se reconfigura el servicio para que el funcionamiento sea el correcto.

4.3.3 Datos de configuración

Para configurar el servicio de enrutamiento es necesario saber hacia dónde vamos a poder enrutar los paquetes. Un destino posible va a ser todas las interfaces WAN, es decir, que tengan configurado un gateway. Como hemos dicho también vamos a tener la posibilidad de enrutar hacia un grupo de interfaces, así que para ellos necesitaremos dejar constancia de cuáles son estos grupos y qué interfaces los componen, así como su modo de funcionamiento, si se va a transmitir por todas las interfaces buscando mayor ancho de banda o se va a trasmitir solo por una en orden de prioridad buscando únicamente redundancia. Los tipos posibles se llamaran Failover y Balancer. Para ello se declara una tabla gateway que contendrá las distintas posibilidades de enrutamiento.

Por otra parte la selección del tráfico y su asignación de un destino de enrutamiento se hará haciendo uso de las reglas del servicio de Firewall, por tanto será necesario incluir campos para referenciar a la interfaz o grupo de balanceo que será por el que se enrute el tráfico (ver figura 12).

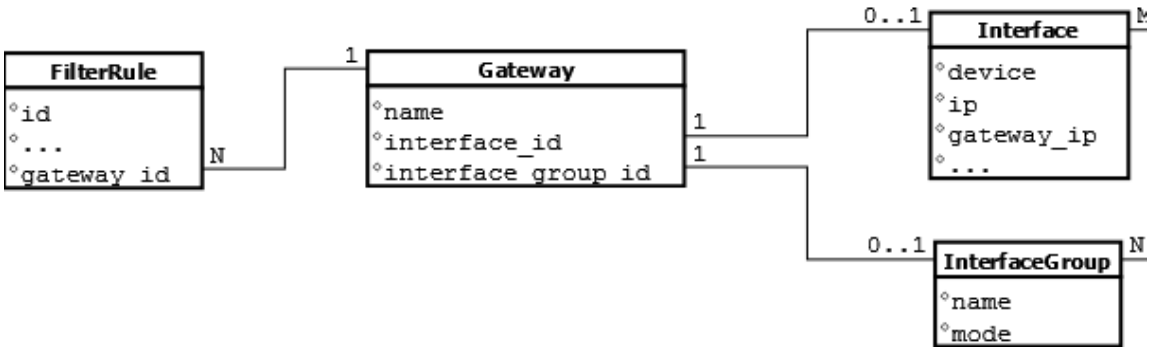

Figura 12: Entidades configuración Routing

4.4 Calidad de Servicio (QoS)

La Calidad de Servicio o Quality of Service (QoS) permite regular el tráfico en una red para garantizar baja latencia o ancho de banda retrasando o descartando paquetes de baja prioridad si se ha sobrepasado la capacidad del enlace. De esta manera podemos hacer una clasificación del tráfico según las prioridades que hayamos determinado, y así conseguiremos que el tráfico que sea más crítico nunca se vea perjudicado por un tráfico menos sensible a latencias más altas. La clasificación de tráfico, en este sistema en particular, se hará mediante las reglas de Firewall explicadas anteriormente. El tráfico será clasificado cuando entre en el sistema y esa clasificación afectará en el momento de su salida. En los siguientes puntos se detalla la solución adoptada y su integración con las reglas de filtrado de paquetes y la base de datos.

4.4.1 QoS en Linux

La calidad de servicio o Quality of Service (QoS) consiste en aplicar ciertas técnicas para garantizar que cierta cantidad de información se transmite a destino en un tiempo determinado. Esto se conoce como el *throughput* de un enlace. QoS es especialmente importante para aplicaciones que tienen requisitos temporales estrictos, como son por ejemplo la transmisión de voz y video en tiempo real (VoIP). Además de ser usado para garantizar cierto caudal de información también se puede utilizar para limitarlo.

Linux implementa QoS mediante un sistema de colas en las que clasificar el tráfico según unos criterios establecidos para cada clase de tráfico. Dichas colas tienen asociada una prioridad y un límite (superior e inferior) de paquetes (o bytes) por unidad de tiempo. Las colas se organizan en forma de árbol, es decir, que una cola puede contener varias colas creando así una jerarquía. Los paquetes serán encolados en una de las hojas del árbol e irán subiendo por el árbol hasta el momento en el que estén en la raíz, que será cuando efectivamente el paquete sea enviado. En la figura 13 se puede ver camino que siguen los paquetes de datos por el sistema QoS.

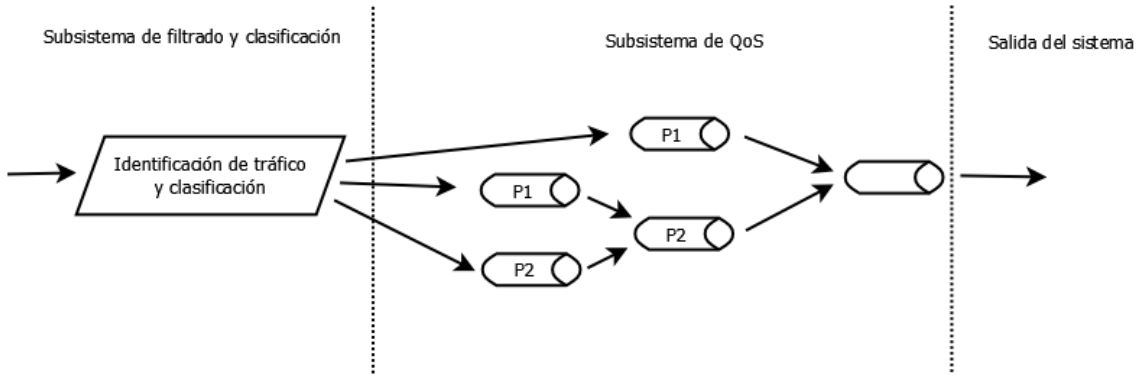

Figura 13: Esquema general de clasificación QoS

Cuando el núcleo del sistema operativo determine que la tarjeta de red está libre para enviar datos lo que hará es desencolar paquetes del nodo raíz y la orden de desencolar se irá transmitiendo por los nodos, teniendo en cuenta siempre la prioridad. Las colas no permiten desencolar paquetes si se ha sobrepasado la tasa máxima especificada y tampoco permitirán encolar paquetes en el mismo caso, descartando dichos paquetes. Esa es la explicación básica del funcionamiento del sistema de colas, lógicamente cada tipo de cola tendrá un funcionamiento interno diferente que la hará más o menos adecuada para cada situación. Hay colas que tienen un comportamiento FIFO (First In First Out) y otras más complejas que son capaces de reordenar los paquetes para proporcionar equidad entre distintas sesiones.

La implementación en Linux presenta un matiz sobre lo anteriormente explicado. Existen dos tipos de colas, según permitan clasificar los paquetes o no. Las colas que no permiten clasificación de paquetes no pueden tener más colas "hijas". En cambio las colas que permiten clasificación de paquetes permiten definir para ello clases a las que sí que es posible asignar otras colas. A cada una de esas clases se le asigna además una prioridad, que permite a la cola determinar de cuál de las colas asociadas a las clases va a desencolar el siguiente paquete a transmitir.

4.4.2 Solución implementada

Las posibilidades que ofrece el subsistema de control de tráfico de Linux son muy amplias y flexibles. Para encontrar la solución más adecuada se han tenido en cuenta los siguientes requerimientos:

- Capacidad para seleccionar un tipo de tráfico y asignarle una velocidad garantizada y un límite de velocidad máximo.
- La selección del tráfico se realizará de la misma manera que las reglas de filtrado de tráfico.
- Los flujos de tráfico de un mismo tipo deben ser tratados con equidad, es decir, que un flujo de tráfico no puede absorber todo el ancho de banda provocando retrasos en otros flujos.

Por ejemplo, teniendo en cuenta todo lo anterior, este sería un caso concreto de uso:

```
Tres tipos de tráfico: Interactivo, Web y Resto
Ancho de banda disponible: 1 Mbit/s
```

```
Asignación mínima y máxima del ancho de banda a cada tipo:
Web: Mínimo 50% Máximo 100%
Interactivo: Mínimo 20% Máximo 100%
Resto de tráfico: Mínimo 5% Máximo 50%
```

Dada la flexibilidad que permite Linux en este tema, las soluciones posibles son varias. Las colas se pueden combinar de cualquier manera formando un árbol, pero en nuestro caso no será necesaria una estructura muy compleja, ya que clasificación del tráfico se hará a un solo nivel, es decir, las clasificaciones de tráfico no contendrán otras subclasificaciones.

4.4.2.1 *Hierarchy Token Bucket*

La disciplina de colas HTB permite clasificación de paquetes y definir una jerarquía de colas, en las que se puede definir individualmente las limitaciones de velocidad del tráfico. La jerarquía comienza definiendo un nodo raíz en el que se define el ancho de banda global. Posteriormente se pueden definir clases que cuelgan del nodo raíz, estableciendo límites de velocidad más específicos. La figura 14 muestra un ejemplo de jerarquía.

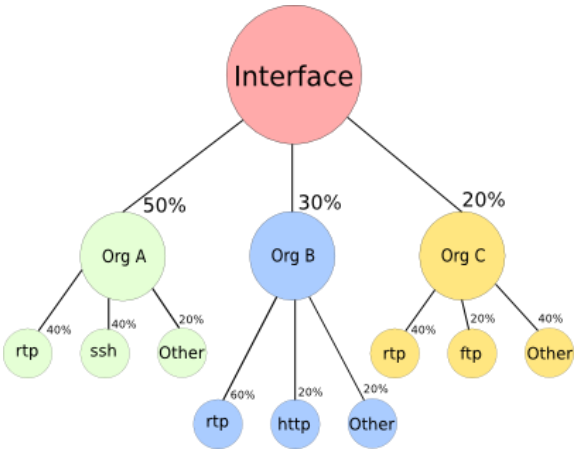

Figura 14: Ejemplo distribución ancho de banda con HTB

Una de las particularidades del algoritmo HTB es que permite establecer velocidades garantizadas para el tráfico. Las subclases de un nodo pueden tomar prestado ancho de banda del nodo superior respetándose siempre el ancho de banda garantizado. Si una clase no está usando su ancho de banda garantizado, este será redistribuido entre las demás clases que lo demanden, volviendo siempre a ser asignado a la primera clase si esta lo demanda. De esta manera se consigue que cada clase tenga su ancho de banda garantizado pero sin que esto sea un desaprovechamiento del ancho de banda si la demanda no llega al mínimo garantizado.

4.4.2.2 *Stochastic Fairness Queueing*

La disciplina de colas SFQ no realiza ningún tipo de control de la velocidad del tráfico, sino que realiza una programación del envío de los paquetes basándose en una clasificación de flujos. Es decir, el algoritmo mantiene información sobre cada flujo de tráfico y programa el envío de cada paquete en función a esa información, de manera que haya una equidad (de ahí su nombre, *fairness*) en el consumo de ancho de banda de todos los flujos e impidiendo que un flujo acapare todo el ancho de banda disponible. De hecho sirve como un medio para mitigar

un ataque de denegación de servicio (DoS). En este contexto entendemos por flujo de tráfico una conexión TCP o UDP, un flujo ICMP, etc. En la figura 15 se pueden observar como SFQ clasifica los flujos.

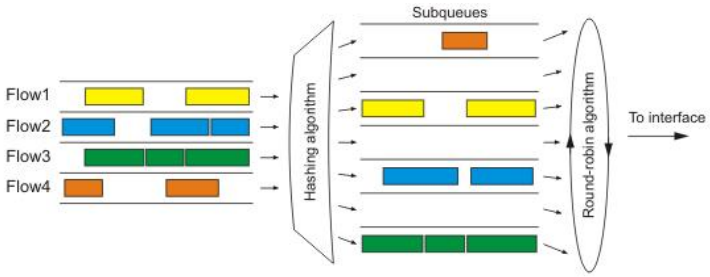

Figura 15: Esquema funcionamiento SFQ

4.4.2.3 *Implementación con HTB y SFQ*

Para cumplir con los requisitos necesarios se ha implementado QoS con una mezcla de las dos disciplinas de colas explicadas anteriormente, HTB y SFQ. HTB nos permitirá garantizar/limitar el ancho de banda y priorizar los distintos tipos de tráfico. SFQ nos permitirá que dentro del mismo tipo de tráfico haya equidad entre todas las sesiones de manera que si una de ellas está usando mucho ancho de banda y el enlace se satura no perjudique a la latencia del resto de sesiones.

La implementación consiste en un nodo raíz de HTB en el que crearemos una clase por cada cola que haya configurada en el sistema, cada una con su prioridad y su ancho de banda. En cada una de esas clases configuraremos una cola SFQ para dentro del mismo tipo de tráfico haya equidad entre sesiones. En la figura 16 se puede ver cómo quedaría la configuración de colas.

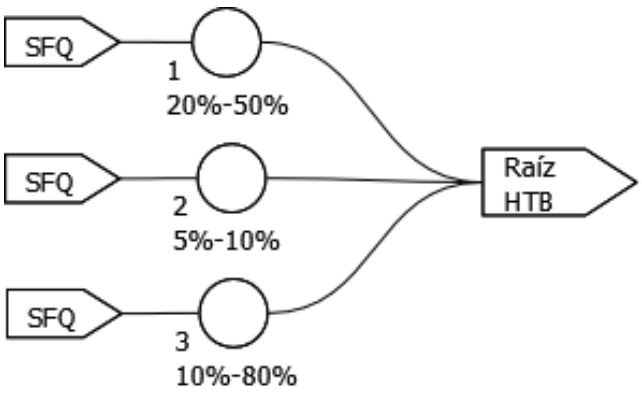

Figura 16: Esquema implementación QoS elegida

4.4.3 Datos de configuración

Como hemos dicho anteriormente la selección y clasificación del tráfico se realizará usando el mismo sistema de reglas utilizado para el Firewall (4.2). Con lo cual lo único que necesitamos es guardar las colas que vamos a utilizar para clasificar el tráfico, cada una con su prioridad y su ancho de banda reservado y limitado. Utilizaremos tantos por ciento en el ancho de banda ya que el ancho de banda de cada interfaz se especificara en su configuración (ver figura 17).

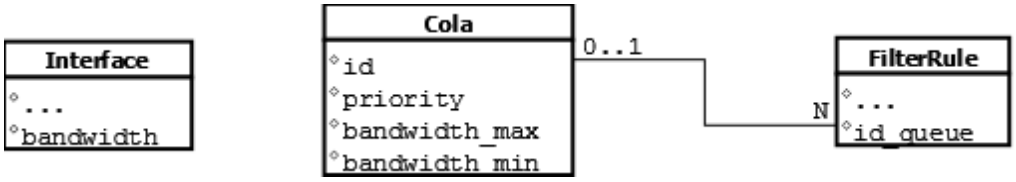

Figura 17: Entidades configuración QoS

4.5 VPN

Las Redes Privadas Virtuales o Virtual Private Networks (VPN) permiten interconectar varias redes, que no tienen conectividad directa, a través de una red intermedia. Normalmente las redes a interconectar serán redes con direccionamiento privado, y la red intermedia será de direccionamiento público. En estas circunstancias es imposible hacer un enrutamiento entra ambas redes sin usar una solución VPN. Las soluciones VPN se basan en encapsular la información en los extremos de las redes privadas y enviarlos a través de paquetes IP en la red pública. De esta forma se consigue que las redes estén interconectadas manteniendo el direccionamiento privado. Además la conexión VPN permite el cifrado de datos, algo que lógicamente es muy importante si se va a trasmitir a través de una red pública.

Este tipo de redes son muy usadas en la actualidad para ahorrar costes en interconexión de sedes de empresas. En vez de contratar un enlace privado con una compañía de telecomunicaciones, lo que se hace es usar internet (red pública) para conectar las distintas sedes.

En la figura 18 presenta un esquema general de una red VPN:

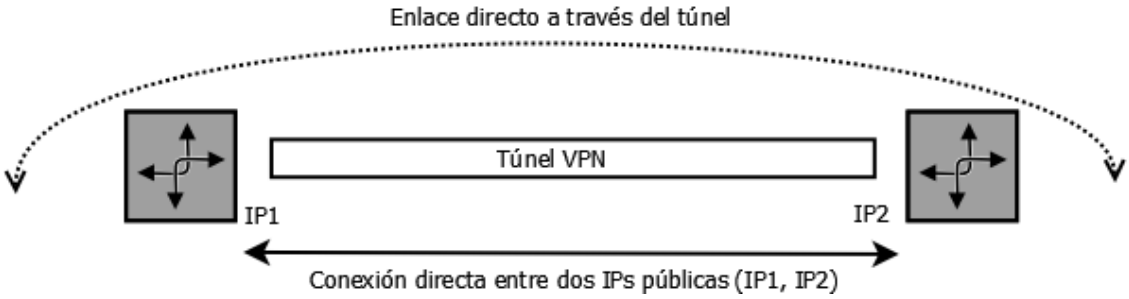

Figura 18: Diagrama general enlace VPN

El túnel VPN se establece entre dos IPs que tienen conectividad directa y todo el tráfico que queramos hacer llegar de un extremo al otro será encapsulado a través del túnel.

4.5.1 Tipos VPN

A grandes rasgos podemos distinguir dos clases de VPN según su uso. Por un lado tenemos las redes VPN para que clientes externos se conecten a una red interna de una organización. Normalmente estas redes se configuran utilizando una infraestructura de clave pública, extendiendo certificados a todos los usuarios de la red y teniendo así control sobre el uso de la misma.

Otro tipo de VPN son las extremo a extremo (*site to site)*, que permiten unir dos redes de una organización en distintas sedes. Como el túnel VPN se establece entre los routers/firewalls en cada sede, el acceso desde los equipos a las demás redes de otras sedes es transparente.

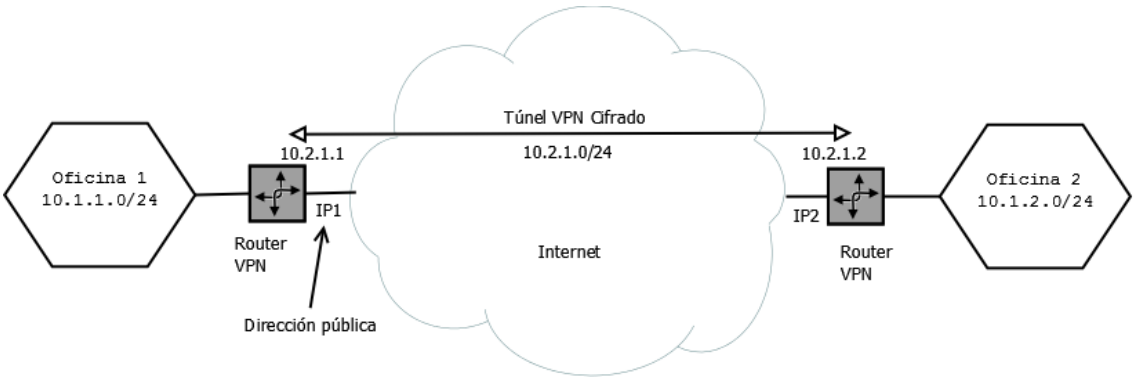

Figura 19: Diagrama enlace VPN site-to-site

La figura 19 muestra un ejemplo de este tipo de VPN. Consiste en dos redes privadas de dos oficinas 10.1.1.0/24 y 10.1.2.0/24 unidas a través de un enlace VPN. Los routers VPN en ambos extremos disponen de dos IPs públicas. Como entre esas IPs hay conectividad directa se puede crear un enlace que servirá para encapsular paquetes de las redes privadas. De esta manera se crea un tunel al que se le ha asignado el rango 10.2.1.0/24 y las IPs 10.2.1.1 y 10.2.1.2 en los extremos. Después de esto lo único necesario para que haya conectividad entre las redes de las oficinas es que ambos routers tendrán que saber qué rangos de direcciones hay en la oficina del otro extremo para poder enrutarlos por el túnel. Es decir, por ejemplo en este caso, el router de la oficina 1 tendrá que saber que los paquetes con destino a 10.1.2.0/24 deberán ser enviados a 10.2.1.2 para que continue el enrutamiento ya dentro de la red de la oficina 2.

En este sistema se han implementado los dos tipos de VPNs y en el caso de las VPN site to site se ha implementado también una técnica para obtener alta disponiblidad en el enalce.

4.5.2 Software empleado

OpenVPN es un software libre que implementa técnicas VPN para crear conexiones punto a punto seguras. Poco a poco se ha ido estableciendo como estándar “de facto” en el mundo de las redes VPN libres.

Como se ha comentado anteriormente en el apartado de interfaces, el fundamento de funcionamiento de este software son las interfaces virtuales que proporciona el sistema operativo. Las interfaces virtuales permiten conectar un proceso de usuario a la interfaz para que este implemente lo necesario para enviar y recibir el tráfico. Es decir, lo que hace un software VPN es escuchar en una interfaz virtual para recibir tráfico y posteriormente encapsularlo sobre una conexión cifrada a través de internet hasta su destino.

4.5.3 Agregación

Como hemos dicho los túneles VPN nos facilitan llegar de una red a otra independientemente de la red intermedia y el camino que tome la información en dicha red. Teniendo en cuenta esto, podemos tener diversos túneles VPN con mismo origen y destino

pero que sean creados sobre distintas redes intermedias. Es decir, podemos tener por ejemplo un sistema con varios enlaces hacia internet y utilizar cada uno de ellos para realizar conexiones VPN. Desde un punto de vista abstracto no importa por qué túnel se encamine los paquetes de datos ya que el resultado en destino será el mismo. Por ello podemos utilizar un conjunto de túneles para balancear el tráfico sobre ellos, aumentando así el ancho de banda y aumentando la disponibilidad ante posibles fallos, como se puede observar en la figura 20.

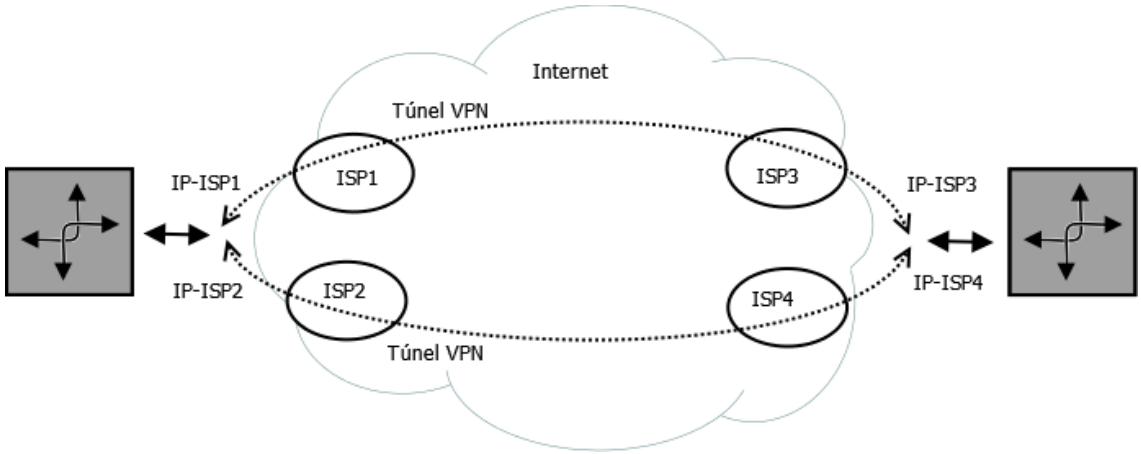

Figura 20: Esquema de agregación de varios enlaces VPN

Para conseguir esta funcionalidad será necesario que el sistema operativo soporte la agregación de interfaces para poder hacer balanceo de la transmisión a nivel de enlace. Como se ha comentado en la sección de interfaces, la agregación de túneles en Linux se puede hacer mediante el driver de bonding. Como los túneles VPN se implementan mediante dispositivos virtuales TAP que a su vez son una emulación de un dispositivo Ethernet regular, entonces es posible añadirlos a una interfaz bonding como si de un dispositivo físico se tratase. También podremos configurar el modo de agregación de la misma manera que con interfaces físicas, es decir, modo de balanceo o modo de tolerancia a fallos. La única restricción es asegurar que todos los túneles añadidos a un bonding tengan como destino la misma máquina y además estén unidos en bonding también en destino, de lo contrario el comportamiento no será el esperado ya que los paquetes acabarán en distintos destinos.

Otra cosa a tener en cuenta es cómo se realizan los enlaces internos, es decir el número de túneles internos y su origen y destino. En nuestro caso se ha optado por crear túneles desde cada IP a todas las del extremo opuesto. Por tanto si tenemos en un extremo N conexiones y en el otro M, tendremos N*M túneles intermedios. Esto nos garantiza alta disponibilidad y tolerancia a fallos siempre que al menos se mantenga operativa la conexión a través de un ISP en cada extremo. Para controlar que túneles están activos se realiza monitorización ARP del otro extremo del túnel mediante el driver de bonding. Si no se reciben respuestas ARP el túnel se considerará inoperativo y se excluirá al hacer la distribución de tráfico, como se puede ver en la figura 21.

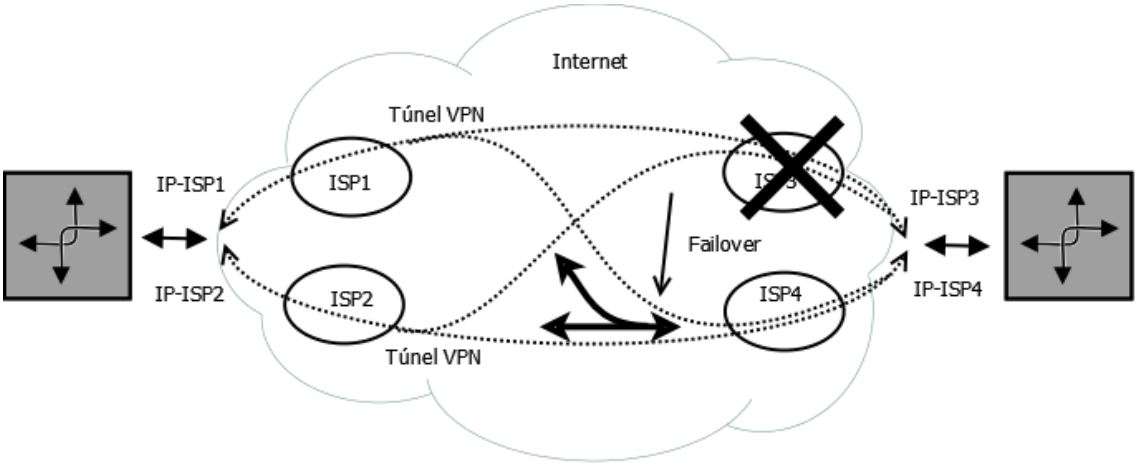

Figura 21: Pérdida de un enlace en VPN agregada.

4.5.4 Datos necesarios en base de datos

Para simplificar el modelo de datos se ha tratado de representar todos los tipos de túneles con una misma entidad, incluso los túneles agregados. En la figura 22 muestra el modelo para representar la configuración VPN.

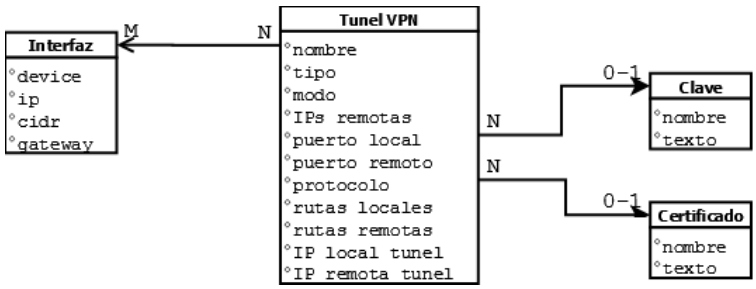

Figura 22: Entidades configuración VPN

La relación con interfaces representa la interfaz o interfaces por las que se enrutará el túnel. Dependiendo del tipo de túnel se interpretará de una forma u otra. Si se trata de un túnel extremo a extremo se comprobará su modo y si es balanceo se usaran todas las interfaces en modo agregado. Si es modo failover se usara la primera que esté activa por orden de prioridad. En cambio sí es un túnel de acceso múltiple se ignora el modo y se toma la primera interfaz activa de las asignadas.

Los demás datos necesarios son los datos de direccionamiento dentro del propio túnel y los puertos e IPs a las que establecer la conexión. También se incluyen referencias a las claves y certificados que harán falta para configurar la seguridad del túnel. De este tema se hablara con más detalle en el apartado de gestión de certificados (4.9.3).

4.6 Control de navegación Web

Una de las funcionalidades del sistema es el control del tráfico de navegación web que atraviesa el sistema. De esta manera podemos tener registros del uso que se hace por parte de los usuarios e incluso establecer restricciones a ciertos tipos de webs.

El control de navegación web se proporciona a través de un servidor proxy que hace de intermediario entre el equipo del usuario en la red local y los servidores reales de las páginas web en internet. Como se ha dicho su principal función, o al menos la que nos interesa en nuestro caso, es la del registro y control del tráfico web. Otra de sus funciones es hacer de caché de contenidos, para acelerar la navegación web, pero es algo que ha ido perdiendo interés con el paso de los años, según se han ido haciendo más rápidos los enlaces a internet y la web se ha ido haciendo cada vez más dinámica.

Existen una gran cantidad de servidores proxy web, aunque se ha seleccionado el más completo y extendido. El software usado como servidor proxy es Squid ya que proporciona todas las funcionalidades requeridas: proporciona registros de navegación, autenticación, filtrado de contenidos y configuración en modo intercepción (transparente). Quizá sea algo pesado para su uso en un sistema empotrado, pero es difícil encontrar otro software capaz de contar con todos los requisitos necesarios.

4.6.1 Tipos

Desde el punto de vista de la interacción del equipo del usuario con el proxy web hay dos tipos bien diferenciados, cada con sus ventajas y sus inconvenientes.

Por un lado tenemos lo que sería un proxy web corriente. Los equipos de los usuarios tienen configurados sus navegadores para que envíen todas las peticiones a través del servidor proxy, del que lógicamente deberán conocer su dirección IP y puerto. Esos datos los han debido obtener alguna fuente, bien sea a través de configuración manual por los técnicos que administren los equipos o a través del protocolo de autodescubrimiento de proxy web WPAD del que se hablará más detalladamente en siguientes puntos. Como inconveniente de este método podemos destacar el tema de la necesidad de configuración en los puestos de usuario, aunque si se automatiza este problema será mínimo. Otro inconveniente es que los usuarios pueden llegar a saber que están navegando a través de un proxy.

El otro tipo de proxy es de intercepción. Los equipos de usuario no conocen la existencia de un proxy en la red, es decir, sus peticiones hacia servidores web en internet se hacen de la manera habitual, como si no existiese proxy. En este caso es el propio firewall el que intercepta las conexiones hacia servidores web en internet y las reenvía hacia el servidor proxy local. La ventaja de este método es que el puesto del usuario no requiere configuración y que el proxy es indetectable por los usuarios. Y como inconveniente destacar que esté método no sirve para interceptar conexiones seguras SSL, ya que el hecho de interceptarlas supone una violación del protocolo y los navegadores lo detectarán como un ataque man-in-the-middle.

4.6.2 Autenticación

Si usamos el proxy normal además podemos añadir autenticación de usuarios. Esto nos permitirá identificar claramente los historiales de navegación de los usuarios de la red. Squid permite autenticar contra muchos tipos de registros, incluso tiene una interfaz genérica para implementar la autenticación a mano. En nuestro caso hemos optado por permitir la autenticación contra un registro local de usuarios, que está almacenado en la propia base de datos de sistema, o contra un servidor LDAP externo.

4.6.3 Autoconfiguración

Para que los navegadores de los puestos de usuario utilicen el servidor necesitan saber su localización (IP y puerto). Como se ha comentado anteriormente la opción de establecerlos manualmente es poco práctica si el número de puestos es numeroso. Para estos casos existe un protocolo llamado WPAD (Web Proxy Autodiscovery Protocol) que está soportado por la mayoría de navegadores web y que permite automatizar este proceso. Este protocolo está definido en el RFC3040 sección 6.4.

El funcionamiento de WPAD se basa por un lado en unos ficheros llamados PAC (Proxy AutoConfiguration) que son ejecutados por el navegador y que contienen una función JavaScript que devuelve la IP y puerto del servidor proxy. Como estos ficheros se ejecutan en el navegador es posible contextualizar la respuesta que se da con variables del puesto del usuario, por ejemplo devolver un servidor proxy u otro en función de la red en la que se encuentra el equipo.

La otra parte del protocolo es la que permite obtener al navegador dichos ficheros PAC. Y aquí presenta dos alternativas para llevarlo a cabo, ambas se basan en proporcionar al navegador una dirección URL de la que descargará el fichero PAC. La primera alternativa hace uso del protocolo DHCP. El navegador lanza una solicitud DHCP especial en la que se pide la dirección URL anterior, el servidor DHCP se la proporciona y el navegador la descarga. La otra alternativa hace uso del DNS. El navegador busca un host llamado "wpad" en la red y una vez recibe la IP trata de descargar el fichero PAC desde una ruta establecida por el protocolo (http://ip/wpad).

El problema de este protocolo está la forma de obtener el fichero PAC ya que unos navegadores soportan el método DHCP (Internet Explorer) mientras que otros sólo soportan el DNS (Mozilla Firefox, Google Chrome, Safari). Esto hace que ambos métodos sean complementarios y haya que implementarlos los dos.

4.6.4 Datos de configuración

Asumiendo que únicamente queremos dar servicio de proxy en las interfaces LAN, lo único que se necesita es saber el tipo de proxy que una interfaz tiene configurado, todo lo demás se puede obtener de la configuración del resto del sistema.

En el caso de que se configure con autenticación sí que necesitamos datos de los usuarios autorizados, para ello se usa una tabla con ese propósito. En la figura 23 se puede ver el modelo.

Figura 23: Entidades configuración proxy web

4.7 Balanceo de tráfico de entrada (DNS)

Como hemos dicho anteriormente es posible que el sistema cuente con varias interfaces que den acceso a internet (interfaces WAN). Con el balanceo de tráfico de entrada se pretende que las peticiones que lleguen desde internet sean distribuidas equitativamente entre todos los enlaces. Como cada enlace tendrá su correspondiente IP pública la única forma que tenemos de distribuir las peticiones entre varias IP es usar DNS.

Para conseguir esta funcionalidad se ha configurado el software PowerDNS de forma que escuche peticiones DNS en las interfaces WAN. Las respuestas a dichas peticiones contendrán siempre las direcciones IP públicas de los enlaces WAN activos (ver punto 4.9.2). De esta manera bastará con configurar el dominio que queramos para que sus servidores DNS apunten a las IP públicas de las interfaces WAN. Si en la resolución DNS resulta que una de las interfaces WAN esta fuera de servicio se pasará a preguntar al siguiente servidor DNS del dominio tal y como especifica el estándar DNS, de esta forma conseguimos que un fallo en una conexión WAN pase desapercibido.

4.8 DHCP y DNS

Este servicio implementa funcionalidad básica para el funcionamiento de una red local como son la asignación de direcciones IP y el servicio de resolución de nombres. Los requisitos que debe cumplir son los siguientes:

- Servidor DHCP en cada interfaz de red.
- Posibilidad de establecer concesiones DHCP estáticas por Hostname, IP y MAC.
- Servidor DNS local para poder registrar nombres locales. Reenvío de peticiones a DNS global si no se encuentra el nombre en local (Forwarding).
- Posibilidad de configurar los servidores DNS para un dominio específico.

Partiendo de estos requerimientos se ha optado por usar el software Dnsmasq. Se trata de un servidor DHCP y forwarder DNS integrado. El hecho de que maneje funciones DHCP y DNS a la vez permite que podamos dar el servicio con un sólo proceso en el sistema, simplificando así su gestión.

4.8.1 Datos de configuración

Según los requisitos comentados en el punto anterior necesitaremos para cada interfaz de red datos para configurar su servidor DHCP. Estos datos son el rango de direcciones que va a ofrecer y el tiempo de validez que tendrán las concesiones que haga (lease time). Para ello hemos añadido a la entidad Interfaz definida en el punto 4.1 los datos anteriores.

Para declarar las concesiones estáticas se ha añadido otra entidad relacionada con la entidad Interfaz. Ésta entidad contendrá Hostname, IP y MAC (ver figura 24).

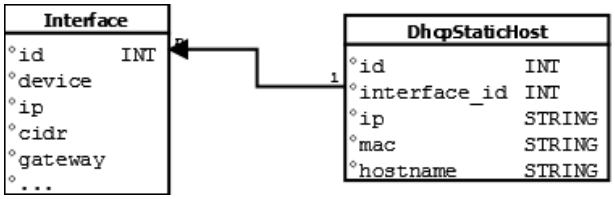

Figura 24: Entidades configuración hosts estáticos DHCP

En cuanto a DNS necesitamos una entidad que guarde los nombres locales y otra que guarde los servidores DNS para dominios específicos (ver figura 25).

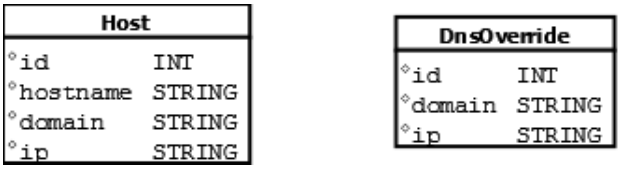

Figura 25: Entidades configuración DNS

4.9 Otros servicios

En este punto se enumeran otros servicios y funcionalidades de las que dispone el sistema.

4.9.1 Análisis y estadísticas de tráfico

Además de todos los servicios que proveen funcionalidad a los equipos de la red, el sistema dispone de varios medios para obtener información sobre el funcionamiento de la red, como por ejemplo cantidad de tráfico a lo largo del tiempo, estado de las conexiones de red, tipo de tráfico en la red, etc.

Estos servicios no influyen directamente sobre el funcionamiento de la red, simplemente permiten tener un mayor control sobre lo que está ocurriendo en ella y así tomar decisiones posteriores para corregir las anomalías detectadas.

Se ha optado por dos software de los mucho que hay disponibles para ello. Por un lado Ntop que permite hacer un análisis del tráfico de red y Collectd que guarda en bases de datos RRD datos sobre el tráfico y rendimiento del sistema para su posterior análisis generando gráficas.

4.9.2 Monitorización de enlaces

Para determinar si un enlace hacia internet está activo se utiliza el programa LSM[2] (Link Status Monitor) que comprueba la disponibilidad de conexión enviando tráfico ICMP hacia una serie de IP definidas. El programa genera un evento cuando se produce un cambio en el estado de los enlaces, de manera que ese evento sea capturado por los servicios que dependan del estado de los enlaces para mantener una configuración óptima.

[2] http://lsm.foobar.fi/

El programa original no permitía dirigir el tráfico de comprobación hacia una interfaz determinada, algo que era clave para monitorizar todas las interfaces independientemente. Para ello se ha modificado el programa para que acepte esta configuración.

4.9.3 Gestión de certificados

Dado que el servicio de VPN hace mucho uso de los certificados se ha implementado una funcionalidad para gestionar y almacenar fácilmente CA, certificados y claves.

El objetivo de la gestión de CA (Certification Authority) y certificados es proporcionar una forma sencilla e integrada de mantener todos los certificados de una CA de una organización. Todos los certificados son guardados en la base de datos de manera que hacer una migración de la configuración sea más simple, en vez de tener que trabajar con todos los certificados en ficheros independientes. Para gestionar una CA se han implementado funciones para inicializar la propia CA, que crean el certificado raíz y el certificado de servidor. Una vez inicializada la CA el trabajo con ella consistirá en dos operaciones, creación y revocación de certificados.

Para implementar los operaciones se ha hecho uso de la infraestructura EasyRSA que viene con el servidor OpenVPN. EasyRSA consiste en una serie de scripts que permiten gestionar una CA que está almacenada en múltiples ficheros dentro de un mismo directorio. Como esto es precisamente lo que no queremos, el trabajo desarrollado ha consistido en aprovechar las partes que nos resultaban útiles de EasyRSA. Es decir, en el uso de sus scripts para generar y revocar certificados e integrando todo lo generado en la base de datos del sistema.

4.9.4 Gestión remota

El sistema implementa un mecanismo de gestión remota que permite administrar el sistema desde fuera de la red privada del dispositivo. Como el sistema se encuentra detrás de un firewall es necesario encontrar la forma óptima de atravesarlo. Una forma de gestión remota podría ser permitir el acceso desde la red externa (internet) al dispositivo; de manera que un agente externo pueda conectar al sistema para administrarlo. Pero esto supone varios problemas, el más importante es que abrir el acceso desde el exterior puede suponer un problema de seguridad si alguien consigue acceder al sistema. El segundo problema tiene que ver con el tipo de dirección en internet que tenga el sistema; si se trata de direcciones IP dinámicas no sabremos en todo momento a que dirección conectar para hacer la gestión remota, tendiendo que proporcionar el cliente la IP a la que conectar.

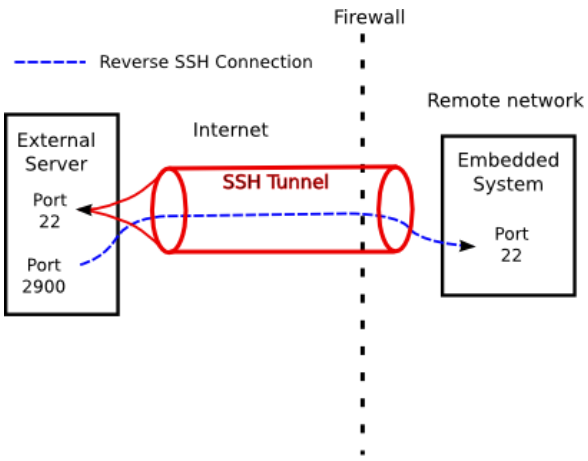

Figura 26: Esquema de gestión remota mediante SSH

Una alternativa es utilizar conexiones inversas, en las que delegamos la iniciativa de conexión al propio sistema, haciendo que se conecte a un sistema remoto del que conocemos perfectamente todos sus datos de acceso. Como el firewall permite todo el tráfico de salida no tendremos problemas en establecer la conexión. Las conexiones inversas consisten en que el sistema detrás del firewall inicia la conexión al sistema externo y una vez se ha establecido el canal de comunicación se usa para tunelizar otras conexiones. Con este sistema conseguimos hacer accesible el sistema desde el exterior de la red local pero manteniendo cerrado el acceso de conexiones nuevas desde el exterior.

Como software para realizar la conexión inversa se ha usado SSH, que aunque su uso mayoritario es para acceso a terminales remotas, también tiene una funcionalidad llamada port forwarding o tunneling, que permite hacer precisamente lo comentado anteriormente. Su funcionamiento se basa en que una vez establecida la conexión segura, el servidor abre un puerto dado para escuchar conexiones. Los datos recibidos en ese puerto son enviados a través de la conexión SSH al cliente, que abre una conexión a la dirección y puerto especificados sobre la que envía los datos recibidos. De la misma manera los datos de vuelta son enviados al servidor a través de la conexión SSH. En la figura 26 se puede observar el túnel SSH.

En este caso se requiere que el sistema guarde los datos de acceso al servidor de administración externo, que serán: IP, puerto y datos de autenticación. De esta forma cuando se requiera hacer una administración remota únicamente habrá que avisar para que sea habilitada. Cuando este habilitada se podrá conectar al servidor de administración en el puerto que haya abierto la conexión de gestión remota. El comando necesario para crear un túnel inverso es el siguiente:

```
ssh -nNT -R 2000:localhost:22 rmgmt@remotehost
```

Con este comando estamos especificando que se conecte a la maquina remota “remotehost” con el usuario “rmgmt”. En esa máquina remota se abrirá un socket de escucha en el puerto 2000. Todas las conexiones a ese socket se dirigirán por el túnel al puerto 22 del propio sistema (localhost).

5 Frontend línea de comandos (CLI)

Aunque inicialmente no contemplado al inicio del proyecto, se ha desarrollado una interfaz de línea de comandos sencilla para poder interactuar con el resto del sistema desarrollado. La motivación ha sido en un principio facilitar las pruebas llevadas a cabo. De esta manera es mucho más fácil introducir la configuración del sistema e ir realizando cambios para comprobar que todos los servicios se configuran correctamente y que todos los eventos que generan son tratados correctamente. Más allá de este propósito, el resultado final ha sido una interfaz de línea de comandos completamente funcional que permite administrar todos los servicios integrados. Por tanto gracias a este Frontend el conjunto del proyecto queda listo para entrar en producción sin ningún desarrollo más.

Se basa en el uso de dos comandos bien diferenciados. El comando "show" nos permitirá mostrar la configuración de los diferentes servicios del sistema, y el comando "config" que nos permitirá modificar la configuración. El principal motivo de esta separación es dejar siempre bien claro que estamos haciendo con el comando que ejecutamos. Si el comando empieza por "show" podremos estar tranquilos de no modificar nada de la configuración, mientras que si empieza por "config" deberemos pensar dos veces lo que estamos haciendo para asegurarnos de que es lo correcto.

A continuación se muestran algunos ejemplos de uso:

```
$ show interfaces
Interface Device     IP              Gateway        Public IP      GwStatus  AdminStatus
=============================================================================================
DMZ         eth4      10.2.1.1                                        -         -
LAN1        br0       10.1.1.2                                        -         -
WAN3        eth2.10   10.1.2.2        10.1.2.1       81.39.190.96     UP        AUTO
WAN         eth1      192.168.1.200   192.168.1.1    81.39.190.96     UP        AUTO

$ show devices
Device      Type        Slaves              MAC Policy     MII Status
=================================================================
br0         Bridge      eth0 eth3           ACCEPT         UP
eth1        Interface   -                   ACCEPT         UP
eth2.10     Vlan        eth2                ACCEPT         UP
eth4        Interface   -                   ACCEPT         UP
virt1       Virtual     -                   ACCEPT         DOWN

$ show interface WAN2 filter
#  If   Action On? Src         Dst       SrcPort  DstPort  Proto  L7Proto Gateway Queue
=============================================================================================
1  WAN2 ACCEPT Yes sshallowed                    22       tcp
2  WAN2 ACCEPT Yes                               vpn      udp
3  WAN2 ACCEPT Yes                                        icmp
4  WAN2 ACCEPT Yes             10.1.1.50                  tcp
```

6 Pruebas realizadas

Dado que el producto desarrollado es un sistema destinado a infraestructura de comunicaciones no ha bastado con desarrollar y probar el sistema sobre la misma máquina en la que va a funcionar. Es necesario simular el entorno que tendrá a su alrededor en la red, para comprobar que toda la funcionalidad de red se comporta adecuadamente. Por ello se ha optado por emplear un entorno de pruebas virtualizado. La virtualización nos permite crear con mucha facilidad todas las máquinas necesarias e incluso definir redes virtuales.

6.1 Entorno virtualizado

Hay múltiples alternativas software que permiten crear un entorno como el comentado anteriormente. En concreto, se ha utilizado VMware ESX por ser el software que se usa para todos los servidores de la empresa.

6.2 Escenarios simulados para llevar a cabo las pruebas

En general la mayoría de servicios se pueden probar con la sola máquina que ejecuta el sistema de enrutamiento, pero hay ciertos casos que requieren simular condiciones que requieren de recursos con los que no se contaba durante el desarrollo. Por ejemplo, todo lo que implica varias conexiones WAN como el balanceo de tráfico de salida y la agregación de conexiones VPN.

Para la prueba de las capacidades de balanceo y redundancia entre varios enlaces WAN se ha usado una configuración con tres máquinas para simular una situación en la que tenemos un sistema con dos enlaces hacia internet. Las otras dos máquinas hacen de pasarela entre internet y el sistema que estamos probando (ver figura 27).

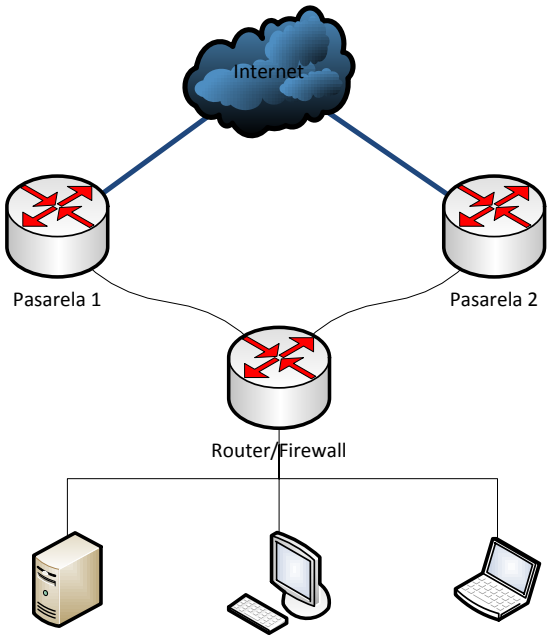

Figura 27: Entorno de pruebas simulado 1

En el caso de la agregación de VPN se ha usado un esquema parecido al anterior pero replicado para tener dos ubicaciones. Para simular la conectividad entre las dos ubicaciones es necesario que las 2 pasarelas tengan conectividad entre ellas, ya que todas las subredes deben ser alcanzables desde cualquiera de ellas. Así simulamos una red que en un caso real es lo que sería Internet (ver figura 28).

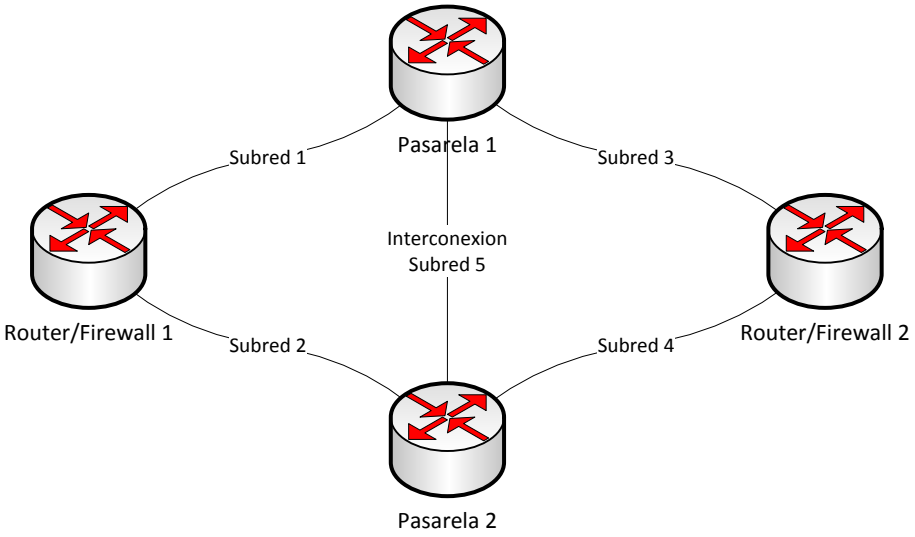

Figura 28: Entorno de pruebas simulado 2

7 Conclusiones

7.1 Grado de cumplimiento de objetivos iniciales

Se han implementado todos los requisitos iniciales del proyecto. Además como se ha comentado en el punto 5 se ha implementado una interfaz de gestión bajo línea de comandos (CLI) lo que permite que el producto sea totalmente funcional.

7.2 Ampliación del trabajo desarrollado

En este punto se presentan las posibilidades de ampliación del trabajo desarrollado.

7.2.1 Interfaz de configuración web

Como ya se ha comentado anteriormente la finalidad de este proyecto es desarrollar un Frontend de gestión web. Dada la arquitectura que se ha usado el desarrollo del Frontend web no deberá preocuparse de nada a nivel de sistema. Simplemente tendrá una interfaz con la base de datos. Esto lo hace mucho más sencillo ya que los desarrolladores no tienen que tener conocimientos específicos sobre todo lo que hace el sistema en sus niveles inferiores.

7.2.2 Consideraciones sobre IPv6

El protocolo IPv6 es la evolución del protocolo IPv4 que actualmente está empezando a sustituir a este. Linux tiene soporte para este protocolo desde hace tiempo así como la mayoría del software empleado. Desde el punto de vista del software desarrollado no es necesario mucho cambio para adaptarse a este protocolo, ya que aunque cambian el formato de las direcciones la configuración de la red a nivel de sistema operativo sigue siendo similar. Únicamente se introducen conceptos nuevos en el tema de asignación de direcciones de red y la ausencia de NAT. La parte que requeriría más cambios es la parte de balanceo de tráfico entre varias interfaces. Recordemos que ésta se basa completamente en NAT. Al no haber NAT en IPv6 no podemos implementar dicha funcionalidad de la misma manera. Actualmente se está desarrollando el RFC6296[3] que estandariza un mecanismo llamado NPT (Network Prefix Translation) que permitirá ofrecer está funcionalidad de forma similar. También existen ya algunas implementaciones experimentales sobre Linux[4].

7.2.3 Incorporación de nuevos servicios

Aunque se han incluido la gestión y configuración de los servicios más comunes en una red, la arquitectura adoptada y la modularidad permiten que añadir nuevos servicios sea una tarea fácil. Asimismo la minimización del acoplamiento entre servicios hace que sea posible personalizar el producto final de acuerdo a las necesidades concretas de cada caso.

7.3 Valoración personal

Una vez finalizado el proyecto valor muy positivamente tanto el proyecto en sí mismo como el hecho de haberlo realizado en una empresa. El ámbito del proyecto me ha permitido adquirir conocimientos de áreas muy distintas, como son las redes de comunicaciones, bases de datos SQL y programación. Volviendo al principio del proyecto diría que ha respondido bien a mis expectativas y que estoy satisfecho con el resultado final.

[3] http://tools.ietf.org/html/rfc6296
[4] http://www.spinics.net/lists/netfilter-devel/msg19979.html

Bibliografía

- Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT and l7-filter – Lucian Gheorghe
- Linux Advanced Routing & Traffic Control: http://lartc.org/
- RFC sobre técnicas multi-homing: http://www.ietf.org/rfc/rfc4116.txt
- RFC sobre DNS load balancing: http://tools.ietf.org/html/rfc1794
- Documentación del proyecto Netfilter/Iptables: http://netfilter.org/documentation/
- Documentación del proyecto iproute2: http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2
- Documentación general sobre Networking en Linux: http://www.linuxfoundation.org/collaborate/workgroups/networking/group
- Documentación del proyecto OpenVPN: http://openvpn.net/index.php/open-source/documentation.html
- Documentación del proyecto PowerDNS: http://doc.powerdns.com/html/index.html
- Documentación del proyecto Squid: http://www.squid-cache.org/Doc/
- Documentación proyecto GNU: http://www.gnu.org/doc/doc.es.html
- Documentación de PHP: http://php.net/manual/es/index.php
- Documentación de Propel: http://propelorm.org/documentation/
- Documentación de SQLite: http://www.sqlite.org/docs.html

A. Anexo Diagrama Entidad-Relación y Modelo Relacional

En este anexo se muestran el Diagrama Entidad-Relación (DER) y su traducción al Modelo Relacional. En la descripción de las tablas del modelo se explica el propósito de cada uno de los atributos de las entidades.

- Diagrama Entidad-Relación (DER)

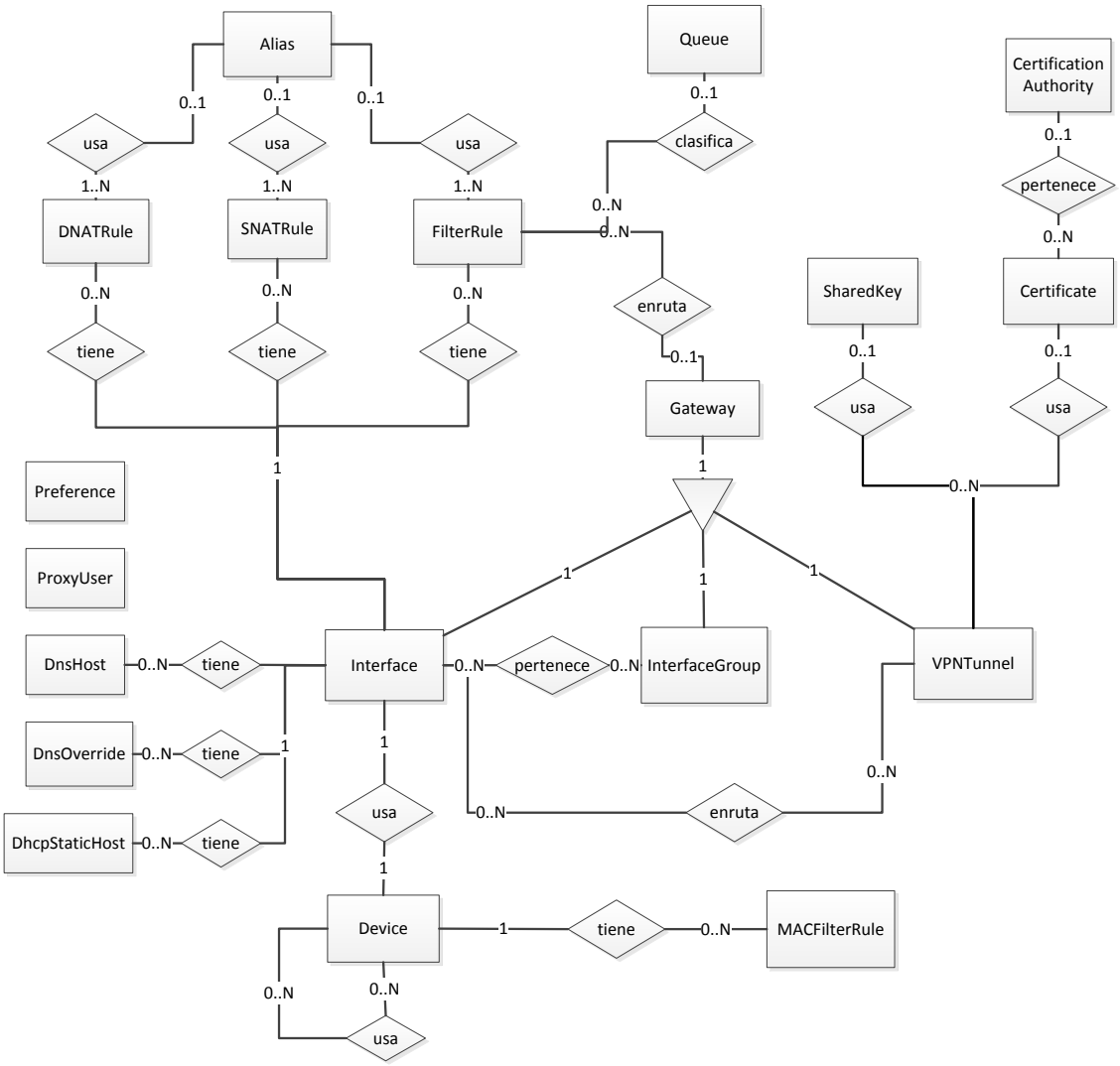

Figura 29: Diagrama Entidad Relación (DER)

- Modelo Relacional

  A continuación se describe la traducción de modelo ER al modelo relacional. Se describen todas las tablas, relaciones y cada uno de sus campos.

  - Tabla gateway:
    - id: integer, clave primaria
    - name: varchar, nombre del gateway
    - device_id: FK(device), device que tiene asociado

        - static_routes: varchar, rutas estáticas asociadas al gateway

- Tabla interface
    - Id: integer, clave primaria.
    - Gateway_id: FK(gateway), gateway del que hereda.
    - Ip: varchar, IP asociada a la interfaz.
    - Cidr: integer, CIDR para generar máscara.
    - public_ip_static: varchar, IP pública estática de acceso a internet, usada por servicio VPN.
    - downbw: integer, Ancho de banda de recepción máximo, usado por servicio QoS.
    - upbw: integer, Ancho de banda de envío máximo, usado por servicio QoS.
    - dhcps_ip_range_start: varchar, ip de inicio del rango dhcp, usado por el servicio DHCP.
    - dhcps_ip_range_end: varchar, ip de fin del rango dhcp, usado por el servicio DHCP.
    - dhcps_lease_time: integer, tiempo de concesión dhcp, usado por el servicio DHCP.
    - dhcps_max_lease_time: integer, tiempo máximo de concesión dhcp, usado por el servicio DHCP.
    - proxy_type: integer, tipo de proxy web que funcionará en la interfaz, usado por el servicio de Proxy Web.
    - proxy_port: integer, puerto en el que funcionará el proxy web, usado por el servicio de Proxy Web.
    - proxy_ldap_ip: varchar, ip del servidor LDAP cuando la autenticación es LDAP, usado por el servicio de Proxy Web.
    - proxy_ldap_base_dn: varchar, DN del LDAP dónde buscar datos de autenticación, usado por el servicio de Proxy Web.
    - admin_status: integer, estado de la interfaz para enrutamiento, 1(AUTO), 2(UP), 3(DOWN). Usado por el servicio de Routing.
    - Clave(name).

- Tabla device
    - Id: integer, clave primaria.
    - Name: varchar, Nombre del dispositivo.
    - Type: integer, Tipo de dispositivo (Interface, Vlan, Bonding, Bridge, Virtual).
    - Master_id: FK(device) referencia a dispositivo master.
    - Slave_id: FK(device) referencia a dispositivo esclavo.
    - Vlan: interger, tag vlan asociada al dispositivo.
    - Bonding_type: integer, tipo de bonding en el caso de que lo sea (ActiveBackup, BalanceRR).
    - Policy: integer, política para el filtrado MAC, (ACCEPT, DROP).
    - Mac: varchar, mac del dispositivo, para sobrescribir la original.
    - Clave(name).

- Tabla monitor_ip
  - Id: integer, clave primaria.
  - Intereface_id: FK(interface), referencia a la interfaz a la que pertenece.
  - Ip: varchar, ip que se monitoriza.
  - Clave(interface_id, ip).

- Tabla interface_group
  - Id: integer, clave primaria.
  - Gateway_id: FK(gateway), gateway del que hereda.
  - Type: integer, tipo de grupo (Balancer, Failover, RoundRobinSticky)
  - Clave(name).
- Tabla interface_interface_group
  - Id: integer, clave primaria.
  - Interface_id: FK(interface), referencia a interface.
  - Interface_group_id: FK(interface_group), referencia a interface_group.
  - Order: integer, orden de preferencia.
  - Clave(interface_id,interface_group_id).
- Tabla alias
  - Id: integer, clave primaria.
  - Name: varchar, nombre del alias.
  - Clave(Name).
- Tabla alias_value
  - Id: integer, clave primaria.
  - Alias_id: FK(alias), alias al que pertenece.
  - Value: varchar, valor del alias.
  - Type: integer, tipo de alias (IP,PORT).
  - Clave(alias_id,vale,type).
- Tabla queue
  - Id: integer, clave primaria.
  - Name: vachar, nombre de la cola.
  - Down_bw_min: integer, % del ancho de banda de recepción mínimo.
  - Down_bw_max: integer, % del ancho de banda de recepción máximo.
  - up_bw_min: integer, % del ancho de banda de envío mínimo.
  - up_bw_max: integer, % del ancho de banda de envío máximo.
  - priority: integer, prioridad de la cola (1-7).
  - Clave(name).
- Tabla filter_rule
  - id: integer, clave primaria.
  - interface_id: FK(interface), interfaz a la que pertenece.
  - disabled: integer, estado de activación (0 desactivada, 1 activada).
  - action: Enum(ACCEPT,DROP,REJECT), acción de la regla.
  - gateway_id: FK(gateway), gateway por el que se enrutará el tráfico.
  - queue_id: FK(queue), cola en la que se encolara el tráfico.
  - protocol_l7: varchar, protocolo de capa de aplicación del tráfico.
  - protocol: varchar, protocolo del tráfico.

- src: varchar, IP origen del tráfico.
- src_port: integer, puerto origen del tráfico.
- dst: varchar, IP destino del tráfico.
- dst_port: integer, puerto destino del tráfico.
- src_alias_id: FK(alias), alias IPs origen del tráfico.
- src_port_alias_id: FK(alias), alias puertos origen del tráfico.
- dst_alias_id: FK(alias), alias IPs destino del tráfico.
- dst_port_alias_id: FK(alias), alias puertos destino del tráfico.
- not_src: integer, inversión del selector de IPs origen (0 no, 1 si).
- not_dst: integer, inversión del selector de IPs destino (0 no, 1 si).
- not_src_port: integer, inversión del selector de puertos origen (0 no, 1 si).
- not_dst_port: integer, inversión del selector de puertos destino (0 no, 1 si).
- order: integer, orden de la regla.

- Tabla dnat_rule
  - id: integer, clave primaria.
  - interface_id: FK(interface), interfaz a la que pertenece.
  - disabled: integer, estado de activación (0 desactivada, 1 activada)
  - protocol: varchar, protocolo del tráfico.
  - ext_ports: varchar, puertos externos a los que afecta la regla.
  - nat_ip: varchar, IP a la que se traduce el destino del tráfico.
  - int_ports: varchar, puertos internos a los que se traduce el tráfico.
  - ext_ports_alias_id: FK(alias), alias puertos externos.
  - int_ports_alias_id: FK(alias), alias puertos internos.
  - fw_rule_auto_add: integer, generación automática de regla de firewall, (0 no, 1 si).
  - order: integer, orden de la regla.

- Tabla snat_rule
  - id: integer, clave primaria.
  - interface_id: FK(interface), interfaz a la que pertenece.
  - disabled: integer, estado de activación (0 desactivada, 1 activada).
  - protocol: varchar, protocolo del tráfico.
  - src: varchar, IP origen del tráfico.
  - src_port: integer, puerto origen del tráfico.
  - dst: varchar, IP destino del tráfico.
  - dst_port: integer, puerto destino del tráfico.
  - src_alias_id: FK(alias), alias IPs origen del tráfico.
  - src_port_alias_id: FK(alias), alias puertos origen del tráfico.
  - dst_alias_id: FK(alias), alias IPs destino del tráfico.
  - dst_port_alias_id: FK(alias), alias puertos destino del tráfico.
  - translation_ip: varchar, IP origen a la que se traduce el tráfico.
  - translation_port: integer, puerto origen al que se traduce el tráfico.

        - static_port: integer, determina si se mantiene el mismo puerto o no, (0 no, 1 si).
        - order: integer, orden de la regla.
    - Tabla vpn_tunnel
        - id: integer, clave primaria.
        - gateway_id: FK(gateway), gateway del que hereda.
        - type: Enum(S2S,PKI_SERVER,PKI_CLIENT), tipo de túnel.
        - dev_type: Enum(TUN,TAP), tipo de dispositivo que se usará.
        - mode: Enum(Balance, Failover), modo de funcionamiento de alta disponibilidad.
        - remote_ips: varchar IPs remotas.
        - local_port: integer, puerto local del túnel.
        - remote_port: integer, puerto remoto del túnel.
        - protocol: Enum(tcp,udp), protocolo del túnel.
        - local_routes: varchar, rutas que van por el túnel.
        - remote_routes: varchar, rutas que se anuncian en el otro extremo.
        - address_pool: varchar, pool de direcciones para servidor PKI.
        - tunnel_local_ip: varchar, ip interna del túnel en modo extremo a exremo.
        - tunnel_remote_ip: varchar, ip interna del túnel en modo extremo a exremo.
        - shared_key_id: FK(shared_key), referencia a clave para cifrado.
        - certificate_id: FK(certificate), referencia a certificado para autenticación y cifrado.
        - autostart: integer, autoarrancar el túnel o no (0 no, 1 si).

    - Tabla interface_vpn_tunnel
        - vpn_tunnel_id, FK(vpn_tunnel), referencia al túnel vpn.
        - interface_id, FK(interface), referencia a la interfaz.
        - order: integer, orden de preferencia de la interfaz.

    - Tabla shared_key
        - id: integer, clave primaria.
        - name: varchar, nombre de la clave.
        - key: varchar, clave.

    - Tabla ca
        - id: integer, clave primaria.
        - name: varchar, nombre de la CA (Certification Authority).
        - crt: varchar, certificado (clave pública y firma).
        - key: varchar, clave privada.
        - crl: varchar, lista de revocación.
        - dh: varchar, parámetros DH.
        - index_txt: varchar, index de EasyRSA.
        - country: varchar, país de la CA.

        - province: varchar, provincia de la CA.
        - city: varchar, ciudad de la CA.
        - organization: varchar, organización de la CA.
        - email: varchar, email del responsable de la CA.

    - Tabla certificate:
        - id: integer, clave primaria.
        - ca_id: FK(ca), ca a la que pertenece el certificado, nulo si no pertenece a ninguna.
        - ca_crt: varchar, certificado de la CA (clave pública y firma).
        - cname: varchar, common name del certificado.
        - name: varchar, nombre del certificado.
        - type: Enum(Client,Server), tipo de certificado.
        - crt: varchar, certificado (clave pública y firma).
        - key: varchar, clave privada.
        - dh: varchar, parámetros DH.
        - crl: varchar, lista de revocación de la CA.
        - is_revoked: integer, estado de revocación (0 ok, 1 revocado).

    - Tabla dns_override
        - id: integer, clave primaria.
        - interface_id: FK(interface), interfaz a la que pertenece.
        - domain: varchar, dominio.
        - ip: varchar, IP del servidor DNS del dominio.

    - Tabla dhcp_static_host
        - id: integer, clave primaria.
        - interface_id: FK(interface), interfaz a la que pertenece.
        - mac: varchar, dirección MAC del host.
        - hostname: varchar, nombre del host.
        - ip: varchar, IP del host.

    - Tabla proxy_user
        - id: integer, clave primaria.
        - interface_id: FK(interface), interfaz a la que pertenece.
        - login: varchar, nombre de usuario.
        - password: varchar, contraseña del usuario.

    - Tabla mac_filter_rule
        - id: integer, clave primaria.
        - device_id: FK(device), dispositivo al que pertenece.
        - mac: varchar, dirección MAC a la que afecta la regla.
        - action: Enum(ACCEPT,DROP), acción de la regla.

    - Tabla host

        - id: integer, clave primaria.
        - name: varchar, nombre del host.
        - domain: varchar, dominio del host.
        - ip: varchar, IP del host.

    - Tabla preferences
        - id: integer, clave primaria.
        - name: varchar, nombre de la opción de configuración.
        - value: varchar, valor de la opción de configuración.

B. Anexo: Proceso de configuración de servicios

En este anexo se describe el proceso de configuración de cada uno de los servicios del sistema. Para cada servicio se detallaran los comandos y ficheros que se generan para que el sistema funcione de la manera descrita en la base de datos.

1. Interfaces

El servicio de configuración de interfaces configura las interfaces una a una empezando por la parte baja del árbol y subiendo hasta que todas han sido configuradas. Cada tipo de interfaz tiene, además de parámetros específicos, una forma distinta de ser configurada.

A continuación se van a detallar todos los comandos que emplea el servicio para configurar las interfaces.

Estado de interfaces

Todas las interfaces, sean del tipo que sean, pueden estar habilitadas o deshabilitadas. Para habilitar o deshabilitar se usan los siguientes comandos:

```
# habilitar
$ ip link set dev eth0 up
# deshabilitar
$ ip link set dev eth0 down
```

Cuando una interfaz está deshabilitada el sistema no enviará ni recibirá tráfico por ella, incluso aunque tenga asignada una configuración IP.

Una vez el sistema se acaba de iniciar podemos obtener una lista de las interfaces disponibles con el siguiente comando:

```
$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
    link/ether 08:00:27:ff:47:d0 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
    link/ether 08:00:27:ff:18:a6 brd ff:ff:ff:ff:ff:ff
```

En este ejemplo podemos observar la interfaz de loopback (lo) siempre presente para comunicaciones internas del sistema, y dos interfaces físicas (eth0 y eth1). Para crear las interfaces virtuales o compuestas se usan distintos comandos en cada caso.

Administración de interfaces bridge o puente

Para administrar las interfaces bridge o puente se utiliza el comando brctl:

```
# creación de bridge entre eth0 y eth1 con el nombre br0
$ brctl addbr br0
$ brctl addif br0 eth0
$ brctl addif br0 eth1
# para borrarlo
$ brctl delbr br0
```

Administración de interfaces agregadas o bonding

Para administrar la agregación de interfaces (bonding) se utiliza la interfaz que presenta el subsistema de bonding del núcleo a través del sistema de ficheros sysfs. Existe un comando para administrarlo pero está prácticamente obsoleto y no soporta muchas de las funcionalidades nuevas del driver. Por ello se ha optado por usar el primer método para configurarlo. Básicamente las operaciones que vamos a usar son crear una interfaz bonding, establecer su tipo y añadir/borrar interfaces físicas de él.

```
# crea el bonding bond0
$ echo +bond0 > /sys/class/net/bonding_masters
# establece el modo de transmisión del bonding. Los modos posibles son:
# balance-rr, active-backup, balance-xor, broadcast,
# 802.3ad, balance-tlb, balance-alb
$ echo active-backup > /sys/class/net/bond0/bonding/mode
# y una vez creado y fijado el tipo de transmisión se añaden interfaces
$ echo +eth0 > /sys/class/net/bond0/bonding/slaves
$ echo +eth1 > /sys/class/net/bond0/bonding/slaves
# y de manera similar se pueden eliminar del bonding (- en vez de +)
$ echo -eth1 > /sys/class/net/bond0/bonding/slaves
```

Administración de interfaces VLAN

En el caso de las interfaces VLAN se usa el comando “ip link” también. Para configurar una interfaz VLAN únicamente necesitamos saber sobre que interfaz queremos configurarla y que identificador de VLAN (tag) queremos asignarle. El nombre que recibe la interfaz se puede elegir libremente, pero por convención en muchos sistemas se forma tomando el nombre de la interfaz concatenada con el tag elegido separado por un punto. Es decir si creamos la interfaz VLAN 2 sobre la interfaz eth0 el nombre quedará “eth0.2”. La forma de crearlas es la siguiente, siguiendo el ejemplo anterior:

```
# creación de la interfaz vlan eth0.2 sobre la interfaz eth0
$ ip link add link eth0 eth0.1 type vlan id 2
# y para borrarla
$ ip link delete eth0.2
```

Administración de interfaces virtuales

Como hemos comentado anteriormente las interfaces virtuales pueden ser de dos tipos. La diferencia son los niveles del modelo OSI que emulan, unas emularán únicamente el nivel de red, mientras que otras emularán también el nivel de enlace. Como en este contexto de configuración de interfaces tenemos posibilidad de mezclarlas y agruparlas (árbol de ejemplo anterior) eso nos limita a usar el segundo tipo nombrado. Para agregar una interface a un bonding o bridge necesitamos que emulen el nivel de enlace. Es por ello que vamos a asumir que todas las interfaces declaradas serán interfaces que emulen el nivel de enlace.

Existen varios comandos para crear y borrar interfaces virtuales. El propio comando "ip link" nombrado anteriormente permite crearlas en sus últimas versiones. Aun así, por no estar disponible en muchas distribuciones, se ha usado el software OpenVPN.

```
# para crear una interface virtual
$ openvpn --mktun --dev tapINTERFAZ
# y para borrarla
$ openvpn --rmtun --dev tapINTERFAZ
```

A la hora de elegir el nombre que se la a la interfaz es importante precederlo de "tap" ya que esto es lo que indica que es una interfaz que emula nivel de enlace. Si quisiésemos una interfaz que emule sólo nivel de red tendríamos que precederlo de "tun".

2. Firewall

En este punto se va a detallar que configuración de iptables se ha usado para implementar las reglas de firewall. Como hemos dicho anteriormente iptables se tiene tres elementos principales: tablas, cadenas y reglas. Aquí veremos que combinación de estas tres nos permite una solución efectiva para nuestro caso.

Expansión y adaptación de reglas invertidas

La creación y modificación de las reglas de filtrado IP en Linux se realiza mediante el comando iptables. El problema de iptables es que trabaja con reglas mucho más sencillas que las almacenadas en la base de datos. La selección de paquetes no se puede hacer en función a un conjunto aleatorio de IPs, solamente respecto a IPs sueltas y rangos consecutivos de IPs. Igualmente ocurre con la selección de paquetes por puerto de origen o destino, únicamente se puede definir un rango o puerto aislado. Además, aunque iptables permite invertir las condiciones de selección de paquetes, tendremos que tener en cuenta al particionar las reglas en reglas simples que no es lo mismo invertir un conjunto de IPs como selección que invertir una a una las IPs del conjunto. Es decir, por ejemplo:

```
Paquetes que no vengan de 1.1.1.1 ó  2.2.2.2 ó 3.3.3.3 rechazar
No es lo mismo que:
Paquetes que no vengan de 1.1.1.1 rechazar
Paquetes que no vengan de 2.2.2.2 rechazar
Paquetes que no vengan de 3.3.3.3 rechazar
```

La primera regla establece que la comunicación desde las tres IPs está permitida, sin embargo, la expansión en reglas sencillas va a rechazar todos los paquetes. Por ejemplo un paquete que venga de 1.1.1.1 pasaría la primera regla, pero sería rechazado en la segunda. Por tanto es necesario expandir las reglas con inversiones de otra manera. En el ejemplo anterior la expansión correcta que mantuviese el comportamiento esperado sería:

```
Paquetes que vengan de 1.1.1.1 aceptar
Paquetes que vengan de 2.2.2.2 aceptar
Paquetes que vengan de 3.3.3.3 aceptar
Resto de paquetes rechazar
```

Extrapolando a un caso general la expansión de reglas con inversiones sería la siguiente:

```
Siendo A, B, C y D conjuntos de IPs y puertos, el signo de interrogación
denotando la inversión y el asterisco denotando cualquier IP o puerto.
Regla original:
Paquetes desde !A a B desde puerto !C a puerto D aceptar
Regla eliminando inversiones:
Paquetes desde A a B desde puerto C a puerto D rechazar
Paquetes desde * a B desde puerto * a puerto D aceptar
```

Por tanto la regla general para pasar una regla con inversiones a una sin inversiones sería que una regla con inversiones se convierte en dos reglas:

```
    1. Sin tener en cuenta las inversiones y acción opuesta
    2. Tomando las inversiones como cualquier IP o puerto y acción de la
regla original
```

De esta manera solucionamos el problema de las inversiones de condiciones y ni siquiera tenemos que usar la negación de condiciones que proporciona iptables.

Reglas filtrado

Para aplicar las reglas de filtrado se utiliza una cadena específica para cada regla. Dentro de cada cadena se realiza la expansión de la regla correspondiente. De esta manera la definición de una regla sería así:

```
    # Añadimos a las cadenas INPUT y FORWARD un salto a
    # la cadena de la regla
    iptables -A INPUT -j RULE{NUM_RULE}
    iptables -A FORWARD -j RULE{NUM_RULE}

    # Añadimos todas las reglas expandidas a la cadena de la regla
    -A RULE{NUM_RULE} -i {INTERFACE} {MATCH1}
     -m state --state NEW -j {ACTION}
    -A RULE{NUM_RULE} -i {INTERFACE} {MATCH2}
     -m state --state NEW -j {ACTION}
```

La parte que aparece como “match” es la que determina las condiciones de aplicación de la regla. Se especifican de la siguiente manera, pudiendo aparecer cualquier combinación de ellas:

```
    Dirección IP Origen: -s IP
    Dirección IP Destino -d IP
    Protocolo: -p PROTOCOLO
    Puerto Origen (TCP, UDP): --sport PUERTO
    Puerto Destino (TCP, UDP): --dport PUERTO
```

Como se observa las reglas solo se aplican a los paquetes que inician una nueva conexión (state es “new”). De esta manera sólo aceptamos tráfico que ha iniciado una sesión correctamente. Para que esto funcione correctamente deberemos indicar que queremos que todo el tráfico con una conexión establecida debe ser aceptado. Para ello añadiremos las siguientes reglas al principio de las cadenas principales:

```
    # En las cadenas INPUT y FORWARD para cada interfaz definida
    iptables -A INPUT -i {INTERFACE} -m state --state ESTABLISHED,RELATED
     -j ACCEPT
    iptables -A FORWARD -i {INTERFACE} -m state --state
     ESTABLISHED,RELATED -j ACCEPT
```

Destination NAT

Las reglas de DNAT (Destination Network Address Translation) permiten hacer traducción de la dirección IP destino de los paquetes. Esto nos permite por ejemplo hacer que los paquetes provenientes de internet sean redirigidos hacia una IP interna.

Las reglas DNAT consisten en un puerto externo y en una IP y puerto interno a los que se redirigen los paquetes haciendo la traducción. De la misma manera que las reglas de filtrado IP, las reglas DNAT también aceptan alias en los puertos internos y externos. Sin embargo, la forma en la que se expanden estas reglas es específica a este tipo y no es igual a las reglas de filtrado. Los alias de puertos de una regla DNAT se expanden haciendo la asignación de puertos uno a uno, por ejemplo:

```
Regla original
Puertos externos: 2222, 2323, 8080
Puertos internos: 22, 23, 80
IP NAT: 192.168.1.100
Regla expandida
Destino puerto 2222 → Traducción a IP 192.168.1.100 puerto 22
Destino puerto 2323 → Traducción a IP 192.168.1.100 puerto 23
Destino puerto 8080 → Traducción a IP 192.168.1.100 puerto 80
```

Si el número de puertos en ambos alias fuese distinto los puertos sobrantes se descartan.

Teniendo en cuenta esto, las reglas DNAT que aplicaremos únicamente tendrán un puerto interno y externo y una IP a la que traducir. Añadiremos las reglas a la cadena PREROUTING de la tabla nat:

```
iptables -t nat  -i {INTERFACE} -p {PROTOCOL} -dport {EXT_PORT} -j
 DNAT {NAT_IP}:{INT_PORT}
```

Source NAT

Las reglas SNAT (Source Network Address Translation) permiten cambiar la dirección origen de los paquetes cuando salgan del sistema. Esto permite que varias IPs de una red privada compartan una única conexión a internet, es decir, una única IP pública.

En este caso las condiciones para seleccionar el tráfico coinciden con las reglas de filtrado, y además se expanden de la misma forma. Una vez tenemos la regla expandida tenemos IP origen y destino, protocolo, puertos y los datos de traducción. Los datos de traducción consisten en la IP y puerto al que traducir. Las reglas se añaden a la cadena POSTROUTING de la tabla nat.

```
# Si tenemos definido puerto de traducción
Iptables -t nat -A POSTROUTING -o {INTERFACE} {MATCH}
 -j SNAT {NAT_IP}:{PORT}
```

```
# Si no tenemos definido Puerto de traducción
Iptables –t nat –A POSTROUTING –o {INTERFACE} {MATCH}
 –j SNAT {NAT_IP} --random
```

Como se puede observar cuando no se establece un puerto de traducción lo que se hace es una traducción a un puerto aleatorio (random). Esto permite evitar el uso de técnicas que permiten establecer una comunicación directa entre dos IP detrás de sus respectivos cortafuegos[5].

Filtrado MAC

En filtrado MAC distinguimos dos casos, cuando el dispositivo al que se referencia es una interfaz normal y cuando la interfaz es un puerto de un bridge. En el primer caso simplemente usaremos *iptables* como se ha explicado anteriormente. Para ello iptables ofrece el módulo de selección por MAC que se configura así:

```
iptables –A FORWARD –i {DEVICE} -m mac --mac-source {MAC} –j {ACTION}
```

En el caso de reglas para un puerto de un *bridge* usamos *ebtables*:

```
ebtables -A FORWARD -i {DEVICE} -s {MAC} -j {ACTION}
```

[5] http://en.wikipedia.org/wiki/TCP_hole_punching

3. Routing

En líneas generales el proceso de configuración consiste en crear una tabla de enrutamiento para cada interfaz WAN. En cada una de esas tablas se asigna como ruta por defecto la ruta hacia el gateway que tiene configurado la interfaz.

```
$ ip route show table WAN1
default via 192.168.1.1 dev eth1
```

Para que el tráfico alcance dichas tablas se le asigna a cada una una regla para que los paquetes marcados con una marca específica vayan a dicha tabla

```
$ ip rule
0:      from all lookup local
1000:   from all lookup main
1001:   from all fwmark 0x1/0xff lookup WAN1
1002:   from all fwmark 0x2/0xff lookup WAN2
1008:   from all fwmark 0x3/0xff lookup WAN3
32767:  from all lookup default
```

Para hacer el marcado de los paquetes se usa el gateway que tienen asignado las reglas del Firewall. Así se marcan dichos paquetes con el ID del Gateway en cuestión:

```
iptables –t mangle -I ROUTINGMARK -j RULE{ID}
iptables –t mangle -A RULE{ID} -i {INTERFAZ} {MATCH}
   -j MARK --set-mark {ID_GATEWAY}/0x00ff
```

En el caso de que el gateway sea un grupo de interfaces tendremos que convertir esa marca a la marca de la interfaz real por la que vamos a enrutar el paquete. Para ello se escriben reglas específicas para cada grupo de interfaces. Si es Failover la marca se convertirá en la marca de la interfaz activa. Si es Balancer se asignara una marca aleatoria entre todas las interfaces activas del grupo

```
# Saltamos a la cadena del grupo si tiene su marca
iptables –t mangle -A BALANCE -m mark --mark 0x{MARK_GROUP}/0xff
 -j GROUP{NAME_GROUP}

# Si es Failover, cambiamos la marca por la de la interfaz activa
iptables –t mangle -A GROUP{NAME_GROUP}
-j MARK --set-mark 0x{MARK_INTERFACE_1}/0xff

# Si es Balancer, aleatoriamente a una interfaz o a otra
iptables –t mangle -A GROUP{NAME_GROUP}
-m mark --mark 0x{MARK_GROUP}/0xff
```

```
-m statistic --mode random --probability 0.33
-j MARK --set-mark 0x{MARK_INTERFACE_1}/0xff
iptables –t mangle -A GROUP{NAME_GROUP}
-m mark --mark 0x{MARK_GROUP}/0xff
-m statistic --mode random --probability 0.5
-j MARK --set-mark 0x{MARK_INTERFACE_2}/0xff
iptables –t mangle -A GROUP{NAME_GROUP}
-m mark --mark 0x{MARK_GROUP}/0xff
-j MARK --set-mark 0x{MARK_INTERFACE_3}/0xff
```

Para conseguir que todos los paquetes de una misma conexión tengan la misma marca de enrutamiento se realiza el marcado únicamente en el primer paquete de la conexión posteriormente se guarda dicha marca en el tracking de conexiones de Netfilter y se restablece cada vez que llega un paquete de la misma conexión.

```
# Restablece marca de conexión.
iptables -t mangle -A PREROUTING
 -j CONNMARK --restore-mark --nfmask 0x0000ffff
# Si el paquete no tiene marca (es 0) lo manda a la cadena de marcado
iptables -t mangle -A PREROUTING
 -m mark --mark 0x000000/0x000000ff -j ROUTINGMARK
# Guardamos la marca a la conexión.
iptables -t mangle -A PREROUTING
-j CONNMARK --save-mark --nfmask 0x0000ffff
```

4. Calidad de Servicio (QoS)

Para implementar la configuración se hace uso de las marcas de paquete que establece el Firewall (ya usadas también para enrutamiento). De esta manera tenemos dos partes, una el marcado de los paquetes al ser identificados y otra la de configuración de las colas y de sus reglas de clasificado del tráfico que repartan los paquetes según la marca que tengan.

En cuanto a la configuración de colas se hace con los siguientes comandos:

```
     # Configuramos el nodo raíz HTB. DEV es el dispositivo y BW ancho de
banda
     tc qdisc add dev DEV root handle 1: htb default 99
     tc class add dev DEV parent 1: classid 1:1 htb rate BW kbit burst 15k

     # Para cada una de las colas se configura:

     # Crea la clase dentro del nodo raíz HTB:
     # ID_COLA es un identificador de la cola.
     # PRIORITY es la prioridad que tiene la cola.
     # BWMIN y BWMAX son los anchos de banda mínimos y máximos calculados
     tc class add dev DEV parent 1:1 classid 1:{ID_COLA}0 htb prio PRIORITY
rate BWMIN kbit ceil BWMAX kbit burst 15

     # Añadimos un nodo SFQ como hijo de la clase HTB
     tc qdisc add dev DEV parent 1:{ID_COLA}0 handle {ID_COLA}00: sfq perturb
10

     # Y finalmente añadimos la regla que hará que los paquetes marcados se
clasifiquen en la cola
     tc filter add dev DEV parent 1:0 prio 0 protocol ip handle
0x{ID_COLA}00/0xff00 fw flowid 1:{ID_COLA}0
```

Para el marcado de los paquetes se hará igual que en enrutamiento aunque usando una máscara distinta para tener marcas de enrutamiento y calidad de servicio diferenciadas.

5. VPN

Como hemos dicho para implementar el servicio VPN se ha usado el software OpenVPN. En este punto se detallara cuáles son las configuraciones necesarias para que dicho software nos de la funcionalidad necesaria. En las configuraciones que requieran agregado de túneles haremos uso de las capacidades de agregación de interfaces (bonding) comentadas en el punto 4.1.

En todos los casos tendremos que tener muy clara la distinción entre direcciones internas del túnel y las direcciones y puertos que permiten que la conexión UDP o TCP subyacente al túnel pueda ser establecida. Estas últimas serán necesarias en todos los casos.

Según las explicaciones anteriores podemos clasificar los tipos de configuraciones que hay que implementar como:

- Acceso múltiple: Servidor.
- Acceso múltiple: Cliente.
- Extremo a extremo: Una interfaz local, una dirección remota.
- Extremos a extremo: Múltiples direcciones remotas y locales.

Para los dos primeros casos de acceso múltiple usaremos las siguientes platillas para configurar OpenVPN:

```
# Servidor para múltiples clientes

local IP_LOCAL     # Dirección IP local en la que escucha el túnel
port 1194          # Puerto en el que escucha
proto udp          # Protocolo del túnel
dev tunserver-adsl # Dispositivo virtual que creará
server 10.6.0.0 255.255.255.0  # Rango de direcciones que asignará a
                               # los clientes
# Certificados
ca /env/firewall/etc/openvpn/server-adsl-ca.crt
cert /env/firewall/etc/openvpn/server-adsl-cert.crt
key /env/firewall/etc/openvpn/server-adsl-cert.key
dh /env/firewall/etc/openvpn/server-adsl-cert.dh

# Rutas locales accesibles para los clientes
push "route 10.1.1.0 255.255.255.0"
```

Para los casos de configuraciones extremo a extremo tendremos el caso básico en el que únicamente queremos establecer un túnel entre una IP local y otra IP remota. Para este caso nos basaremos en esta plantilla:

```
local 192.168.1.200    # Dirección IP local en la que escucha el túnel
remote  199.180.255.14  # Dirección IP remota del túnel
lport 502   # Puerto del extremo local del túnel
```

```
rport 502    # Puerto del extremo remote del túnel
proto udp    # Protocolo de la conexión
dev tapusabr  # Dispositivo virtual
secret /env/firewall/etc/openvpn/usabr.key  # Clave de cifrado
```

En el caso de múltiples interfaces locales y/o múltiples direcciones remotas será necesaria una configuración agregada. En decir, en este caso vamos a tener múltiples instancias de OpenVPN cada una con su configuración y su dispositivo virtual. Para agregar esos túneles intermedios crearemos una interfaz bonding a la que añadiremos todos esos dispositivos virtuales como esclavos. De esta manera conseguimos que la interfaz bonding se convierta de forma transparente para el resto del sistema en el extremo del túnel agregado.

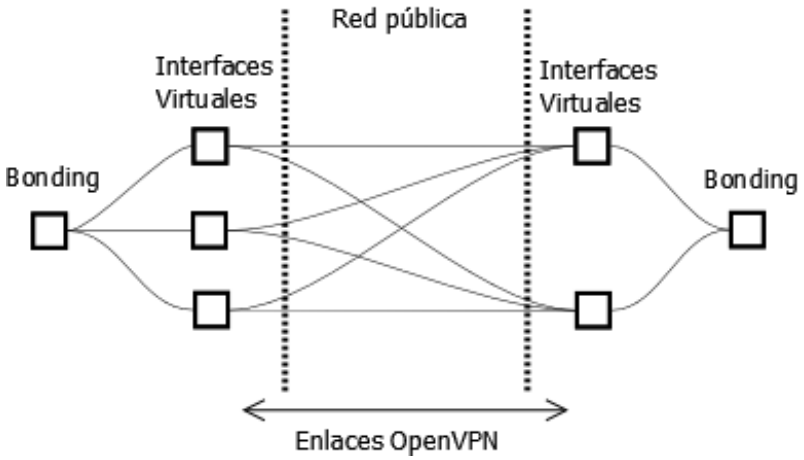

Figura 30: Enlaces generados en una VPN agregada

A la hora de configurar es necesario hacer una asignación de puertos para cada uno de los enlaces ya que vamos a tener varias conexiones sobre una misma IP lo que hace imposible que todas usen el mismo rango de puertos. Esto se ha solucionado estableciendo el puerto de inicio en la configuración y automáticamente se hace una asignación en función del número de enlaces necesarios (ver figura 30).

La configuración de las interfaces bonding se realiza como en el punto 4.1.3 y la única diferencia será que aquí configuraremos la monitorización ARP que proporciona el driver de bonding para que los sub-túneles inoperativos sean excluidos al distribuir el tráfico:

```
# Añadimos el dispositivo "vpnbond" como bonding
echo +vpnbond > /sys/class/net/bonding_masters
# Configuramos la monitorización ARP con la ip del extremo del túnel
echo 1000 > /sys/class/net/vpnbond/bonding/arp_interval
echo +10.8.1.2 > /sys/class/net/vpnbond/bonding/arp_ip_target
```

Para cada sub-túnel se realiza la configuración como en el caso simple con los puertos asignados a cada uno de ellos como se ha explicado anteriormente.

Estas configuraciones se basan en el estado de conectividad de las interfaces en el momento de configuración, pero es necesaria una configuración dinámica atendiendo al estado de conectividad. Las configuraciones son regeneradas cuando una de las interfaces del sistema pierde la conexión o la recupera. Aquí es donde entra la monitorización de interfaces explicada en el punto 4.9.2. Cuando una interfaz pierde o recupera la conexión se generan un evento que el servicio VPN intercepta para comprobar si la configuración de los túneles es la adecuada o se necesitan hacer cambios. Será necesario regenerar la configuración en los casos de túneles que tengan asignada más de una interfaz de enrutamiento y funcionen en modo failover, ya que la interfaz por la que estaban configurados ha podido perder la conectividad o por el contrario una interfaz con más prioridad que la actual ha recuperado la conectividad.

6. Control de navegación web

Partiendo de los datos de configuración en base de datos vamos a tener que el servicio de Proxy puede prestar servicio a 1 o más interfaces de red y pudiendo ser diferente el tipo de funcionamiento para cada interfaz. Es decir, podemos tener dos interfaces LAN, una funcionando en modo intercepción y otra en modo normal con autenticación. Para conseguir este comportamiento se ha usado los ACL (Access Control List) que proporciona Squid. De esta manera configuraremos el servicio y luego diremos qué condiciones se deben cumplir desde cada rango de direcciones (cada LAN) desde las que llegan peticiones. El fichero de configuración quedaría así:

```
# Configuraciones fijas
acl all src all
acl localhost src 127.0.0.1
follow_x_forwarded_for allow localhost
access_log LOG_FILE squid
pid_filename PID_FILE

# Configuración de la autenticación, se crea el ACL "auth"
# que se usara en las LAN que lo requieran
auth_param basic program /usr/lib/squid/ncsa_auth USERS_FILE
acl auth proxy_auth REQUIRED

# Para cada LAN creamos un acl con su rango de IPs
acl LAN1 src 10.1.1.0/24
acl LAN2 src 10.1.2.0/24

# Modo de acceso para cada LAN
http_access allow LAN1
http_access allow LAN2 auth

# Puerto en el que escucha el proxy
# y habilitar modo transparente o intercepción
http_port 127.0.0.1:3128 transparent
```

Adicionalmente si el modo de funcionamiento es intercepción tendremos que configurar una regla en el Firewall para redirigir a Squid todo el tráfico con destino al puerto 80

```
iptables -t nat -A PROXY -i DEVICE_LAN -p tcp --dport 80 -j REDIRECT --
to-ports 3128
```

7. DHCP y DNS

Como hemos dicho anteriormente, Dnsmasq proporciona todos estos servicios mediante un único proceso y su configuración. Por tanto la configuración del servicio consistirá en generar dicho fichero y lanzar el proceso.

A continuación se van a detallar el formato y contenido del fichero de configuración. Cada línea del fichero representa una opción de configuración. Una parte del fichero son opciones fijas y otras son generadas consultando el contenido de la base de datos definido en el punto anterior.

```
bind-interfaces
domain=home
no-resolv
localise-queries
dhcp-leasefile=/env/firewall/run/dnsmasq.leases
dhcp-script=/opt/firewall/backend/bin/dhcp-event
server=8.8.8.8
server=8.8.4.4

interface=br0
dhcp-range=br0,10.1.1.100,10.1.1.199,3600

dhcp-host=a0:0b:ba:a8:b3:cd,10.1.1.120,pc1
dhcp-host=c8:df:7c:09:29:a3,10.1.1.121
```